**Naval Research Laboratory**
**Stennis Space Center, MS 39529-5004**

NRL/MR/7320--02-8285

# Software Design Description for the Simulating WAves Nearshore Model (SWAN)

Richard Allard
Erick Rogers

*Ocean Dynamics and Prediction Branch*
*Oceanography Division*

Suzanne N. Carroll
Kate V. Rushing

*Planning Systems Incorporated*
*Stennis Space Center, MS*

November 15, 2002

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



**1. REPORT DATE** *(DD-MM-YYYY)*
November 15, 2002

**2. REPORT TYPE**
Memorandum

**3. DATES COVERED** *(From - To)*

**4. TITLE AND SUBTITLE**

Software Design Description for the Simulating WAves Nearshore Model (SWAN)

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**
0602435N

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**6. AUTHOR(S)**

Richard Allard, Erick Rogers, Suzanne N. Carroll,* and Kate V. Rushing*

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Naval Research Laboratory
Oceanography Division
Stennis Space Center, MS 39529-5004

**8. PERFORMING ORGANIZATION REPORT NUMBER**

NRL/MR/7320--02-8285

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

**10. SPONSOR / MONITOR'S ACRONYM(S)**

**11. SPONSOR / MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**

*Planning Systems Incorporated, Stennis Space Center, MS 39529-5004

**14. ABSTRACT**

Simulating WAves Nearshore (SWAN) is a third-generation numerical wave model developed for wave computations in coastal regions and inland waters. The model is based on an Eulerian formulation of the discrete spectral balance of action density that accounts for refractive propagation over arbitrary bathymetry and current fields. SWAN is driven by boundary conditions and local winds. The processes of wind generation, whitecapping, quadruplet wave-wave interactions, bottom dissipation, triad wave-wave interactions, and depth-induced wave breaking are represented explicitly, though SWAN does not account for diffraction. SWAN's numerical propagation scheme is implicit; thus the model is most efficient (relative to other models) when applied to cases with relatively high geographic resolution (i.e., cases of smaller scale). SWAN has been validated by comparisons with analytical solutions, and laboratory and field observations.

SWAN is the state-of-the art phase-averaged coastal wave model (at the time of this writing). As a third-generation model, SWAN models propagation and dissipation explicitly. It also allows for simple integration of future developments in formulations for the physical processes mentioned above, as SWAN is a strictly and logically modular program.

**15. SUBJECT TERMS**

SWAN, WAM, Wavewatch III, Subroutines, Waves

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| **a. REPORT** | **b. ABSTRACT** | **c. THIS PAGE** | | | Richard Allard |
| Unclassified | Unclassified | Unclassified | UL | 219 | **19b. TELEPHONE NUMBER** *(include area code)* (228) 688-4894 |

**Standard Form 298 (Rev. 8-98)**
**Prescribed by ANSI Std. Z39.18**

# Software Design Description

# for the

# Simulating WAves Nearshore Model (SWAN)

Cycle III Version 40.11

Contract Number
N00014-96-D-6031

March 2002

Richard Allard
Erick Rogers

Ocean Dynamics and Prediction Branch
Oceanography Division
Naval Research Laboratory

Suzanne N. Carroll
Kate V. Rushing
Planning Systems Incorporated

PSI Technical Report SSC-002-02

# TABLE OF CONTENTS

# LIST OF FIGURES



# LIST OF TABLES

# 1.0 SCOPE

## 1.1 INTRODUCTION

Simulating WAves Nearshore (SWAN) is a third-generation numerical wave model developed for wave computations in coastal regions and inland waters. The model is based on an Eulerian formulation of the discrete spectral balance of action density that accounts for refractive propagation over arbitrary bathymetry and current fields. SWAN is driven by boundary conditions and local winds. The processes of wind generation, whitecapping, quadruplet wave-wave interactions, bottom dissipation, triad wave-wave interactions and depth-induced wave breaking are represented explicitly, though SWAN does not account for diffraction. SWAN's numerical propagation scheme is implicit; thus the model is most efficient (relative to other models) when applied to cases with relatively high geographic resolution (i.e. cases of smaller scale). SWAN has been validated by comparisons with analytical solutions and laboratory and field observations.

SWAN is the state of the art phase-averaged coastal wave model (at the time of this writing). As a third generation model, SWAN models propagation and dissipation explicitly. It also allows for simple integration of future developments in formulations for the physical processes mentioned above, as SWAN is a strictly and logically modular program.

## 1.2 DOCUMENT OVERVIEW

The purpose of this Software Design Description (SDD) is to describe the software design and code of the Simulating WAves Nearshore model (SWAN). The SDD gives a summary of model operations, physics and basic equations and a description of source code components. Most importantly, the SDD gives a detailed description of the source code components, such as subroutines and common blocks, which make up the SWAN model.

Manuscript approved August 29, 2002.

# 2.0 REFERENCE DOCUMENTS

## 2.1 SWAN SOFTWARE DOCUMENTATION

Carroll, S., Kelly, K. (2002). "User's Manual for the Simulating WAves Nearshore Model (SWAN) Cycle III Version 40.11." PSI Technical Report SSC-001-02.

Holthuijsen, L. H., Booij, N., Ris, R. C., Haagsma, IJ. G., Kieftenburg, A. T. M. M., and Kriezi, E. (2000). "SWAN Cycle III Version 40.11 User Manual, Electronic Version." Delft University of Technology, the Netherlands.

## 2.2 SWAN SOFTWARE RELEASE

Booij, N., Haagsma, IJ. G., Kieftenburg, A. T. M. M., and Holthuijsen, L. H. (2000). "SWAN Cycle II Version 40.11 Implementation Manual, Unauthorized Electronic Version." Delft University of Technology, the Netherlands.

Holthuijsen, L. H., Booij, N., Ris, R. C., Haagsma, IJ. G., Kieftenburg, A. T. M. M., and Kriezi, E. (2000). "SWAN Cycle III Version 40.11 User Manual, Unauthorized Electronic Version." Delft University of Technology, the Netherlands.

## 2.3 GENERAL TECHNICAL DOCUMENTATION

Abbott, M. B. and Basco, D. R., (1989). Computational Fluid Dynamics. John Wiley and Sons Inc., New York, p. 425.

Abreu, M., Larraza, A., and Thornton, E., (1992). Nonlinear transformation of directional wave spectra in shallow water. *J. Geophys. Res.*, 97: 15579-15589.

Arcilla, A. S., Roelvink, J. A., O'Connor, B. A., Reniers, A. J. H. M., and Jimenez, J. A., (1994). The Delta flume '93 experiment. *In the Proc. of the Coastal Dynamics Conf.*, 488-502.

Arcilla, A. S. and Lemos, C. M., (1990). Surf-zone Hydrodynamics. *Centro Internacional de Métodos Numéricos en Ingenieria*, Barcelona, Spain, 310 pp.

Banner, M. L. and Young, I. R., (1994). Modeling spectral dissipation in the evolution of wind waves, Part I: Assessment of existing model performance. *J. Phys. Oceanogr.*, 24(7): 1550-1571.

Battjes, J. A. and Beji, S., (1992). Breaking waves propagating over a shoal. *Proc. of 23rd Int. Conf. Coastal Engineering*, ASCE, 42-50.

Battjes, J. A. and Stive, M. J. F., (1985). Calibration and verification of dissipation model for random breaking waves. *J. Geophys. Res.*, 90(C5): 9159-9167.

Battjes, J. A. and Janssen, J. P. F. M., (1978). Energy loss and set-up due to breaking of random waves. *Proc. of 16th Int. Conf. on Coastal Eng.*, ASCE, New York, 569-587.

Beji, S. and Battjes, J. A., (1993). Experimental investigation of wave propagation over a bar. *Coastal Eng.*, 19: 151-162.

Bertotti, L. and Cavaleri, L., (1994). Accuracy of wind and wave evaluation in coastal regions. *Proc. of 24th Int. Conf. Coastal Eng.*, ASCE, 57-67.

Booij, N., Ris, R. C., and Holthuijsen, L. H., (1999). A third-generation wave model for coastal region, Part I: Model description and validation. *J. Geophys. Res.,* 104(C4): 7649-7666.

Booij, N., Ris, R. C., and Holthuijsen, L. H., (1999). A third-generation wave model for coastal region, Part II: Verification. *J. Geophys. Res.,* 104(C4): 7667-7681.

Booij, N., Holthuijsen, L. H., and Ris, R. C., (1993). A spectral wave model for the coastal zone. In: *Proc. of the 2nd Int. Symp. on Ocean Wave Meas. and Analysis*, New Orleans, LA, New York pp. 630-641.

Booij, N. and Holthuijsen, L. H., (1987). Propagation of ocean waves in discrete spectral wave models. *J. of Comp. Phys.*, 68: 307-326.

Bouws, E. and Komen, G. J., (1983). On the balance between growth and dissipation in an extreme, depth-limited wind-sea in the southern North Sea. *J. Phys. Oceanogr.*, 13: 1653-1658.

Cavaleri, L. and Malanotte-Rizzoli, P., (1981). Wind wave prediction in shallow water: Theory and applications. *J. Geophys. Res.*, 86(C11): 10961-10973.

Chen, Y. and Guza, R. T., (1997). Modeling of breaking surface waves in shallow water, *J. Geophys. Res.*, 102(C11): 25035-25046.

Collins, J. I., (1972). Prediction of shallow water spectra. *J. Geophys. Res.*, 77(15): 2693-2707.

Dingemans, M. W., (1997). Water wave propagation over uneven bottoms, Part 1: Linear wave propagation. *Adv. Ser. on Ocean Eng.*, 13, World Scientific, 471 pp.

Dingemans, M. W., Radder, A. C. and DeVriend, H. H., (1987). Computation of the driving forces of the wave-induced currents. *Coastal Eng.* 11: 539-563.

Eldeberky, Y., (1996). Nonlinear transformation of wave spectra in the nearshore zone, *Ph.D. thesis*. Delft University of Technology, Department of Civil Engineering, Delft, Netherlands.

Eldeberky, Y. and Battjes, J. A., (1996). Spectral modeling of wave breaking: Application to Boussinesq equations. *J. Geophys. Res.*, 101(C1): 1253-1264.

Eldeberky, Y. and Battjes, J. A., (1995). Parameterization of triad interactions in wave energy models. *Proc. Coastal Dynamics Conf.*, Gdansk, Poland, 140-148.

Elgar, S., Guza, R. T., Raubenheimer, B., Herbers, T. H. C., and Gallagher, E. L., (1997). Spectral evolution of shoaling and breaking waves on a barred beach. *J. Geophys. Res.*, 102(C7): 15797-15805.

Fletcher, C. A. J., (1988). Computational Techniques for Fluid Dynamics, Parts I and II, Springer-Verlag, Berlin; New York, pp. 409-484.

Galvin, C. J., (1972). Wave Breaking in Shallow Water, Waves on Beaches and Resulting Sediment Transport, Academic Press Inc., San Diego, California, pp. 413-455.

Goda, Y., Takeda, H., and Moriya, Y., (1967). Laboratory investigation of wave transmission over breakwaters. *Rep. Port and Harbour Res. Inst.*, 13 (from Seelig, 1979).

Golub, G. H. and Van Loan, C. F., (1986). Matrix Computations. Academic Press, London, p. 476.

Günther, H., Hasselmann, S., and Janssen, P. A. E. M., (1992). "The WAM model Cycle 4 (Revised Version)." Deutsch. Klim. Rechenzentrum, Technical Report No. 4, Hamburg, Germany.

Hasselmann, S. and Hasselmann, K., (1985). Computations and parameterizations of the nonlinear energy transfer in a gravity wave spectrum. Part I: A new method for efficient computations of the exact nonlinear transfer integral. *J. Phys. Oceanogr.* 15: 1369-1377.

Hasselmann, S., Hasselmann, K., Allender, J. H., and Barnett, T. P., (1985). Computations and parameterizations of the nonlinear energy transfer in a gravity wave spectrum, Part II: Parameterizations of the nonlinear transfer for application in wave models. *J. Phys. Oceanogr.*, 15(11): 1378-1391.

Hasselmann, S. and Hasselmann, K., (1981). A symmetrical method of computing the non-linear transfer in a gravity-wave spectrum. *Geophys. Einzelschr., Ser. A.*, Geophys. Inst., Univ. of Hamburg, Hamburg, Germany, 52(8).

Hasselmann, K., (1974). On the spectral dissipation of ocean waves due to whitecapping. *Boundary-layer Meteor.*, 6: 1-2, 107-127.

Hasselmann, K., Barnett, T. P., Bouws, E., Carlson, H., Cartwright, D. E., Enke, K., Ewing, J. A., Gienapp, H., Hasselmann, D. E., Kruseman, P., Meerbrug, A., Muller, P., Olbers, D. J., Richter, K., Sell, W., and Walden, H., (1973). Measurements of wind-wave growth and swell decay during the JOint North Sea WAve Project (JONSWAP). *Dtsch. Hydrogr. Z.*, 12(A80): 95.

Hasselmann, K. and Collins, J. I., (1968). Spectral dissipation of finite-depth gravity waves due to turbulent bottom friction. *J. Mar. Res.*, 29: 1-12.

Holthuijsen, L. H., Booij, N., and Herbers, T. H. C., (1989). A prediction model for stationary, short-crested waves in shallow water with ambient currents. *Coastal Eng.*, 13: 23-54.

Holthuijsen, L. H. and De Boer, S., (1988). "Wave forecasting for moving and stationary targets." Computer Modeling in Ocean Engineering. B.Y. Schrefler and O.C. Zienkiewicz, eds., Balkema Publishing Co., Rotterdam, Netherlands, pp. 231-234.

Janssen, P. A. E. M., (1992). Experimental evidence of the effect of surface waves on the airflow. *J. Phys. Oceanogr.* 22:1600-1604.

Janssen, P. A. E. M., (1991a). Quasi-linear theory of wind-wave generation applied to wave forecasting. *J. Phys. Oceanogr.*, 21: 1631-1642.

Janssen, P. A. E. M., (1991b). Consequences of the effect of surface gravity waves on the mean air flow, paper presented at the *Breaking Waves Int. Union of Theor. And Appl. Mech. (IUTAM)*, Sydney Australia, 193-198.

Janssen, P. A. E. M., (1989). Wave induced stress and the drag of air flow over sea waves. *J. Phys. Oceanogr.*, 19: 745-754.

Jonsson, I. G., (1980). A new approach to rough turbulent boundary layers. *Ocean Eng.*, 7: 109-152.

Jonsson, I. G. and Carlsen, N. A., (1976). Experimental and theoretical investigations in an oscillatory turbulent boundary layer. *J. Hydraulic Res.*, 14: 45-60.

Jonsson, I. G., (1966). Wave boundary layers and friction factors. *Proc. of 10th Int. Conf. Coastal Eng.*, ASCE, 127-148.

Kaminsky, G. M. and Kraus, N. C., (1993). Evaluation of depth-limited wave breaking criteria. *Proc. of 2nd Int. Sym. on Ocean Wave Meas. and Analysis*, New Orleans, Louisiana, 180-193.

Komen, G. J., Cavaleri, L., Donelan, M., Hasselmann, K., Hasselmann, S., and Janssen, P. A. E. M., (1994). Dynamics and Modeling of Ocean Waves. Cambridge Univ. Press, New York, p. 532.

Komen, G. J., Hasselmann, S., and Hasselmann, K., (1984). On the existence of a fully developed wind-sea spectrum. *J. Phys. Oceanogr.*, 14: 1271-1285.

Longuet-Higgins, M. S., (1969). On wave breaking and the equilibrium spectrum of wind-generated waves. *Proc. of Roy. Soc. A.* 310: 151-159.

Luo, W. and Monbaliu, J., (1994). Effects of the bottom friction formulation on the energy balance for gravity waves in shallow water. *J. Geophys. Res.*, 99(C9): 18501-18511.

Madsen, P. A. and Sørensen, O. R., (1993). Bound waves and triad interactions in shallow water. *Ocean Eng.,* 20(4): 359-388.

Madsen, O. S., Poon, Y. K., and Graber, H. C., (1988). Spectral wave attenuation by bottom friction: Theory. *Proc. of 21st Int. Conf. Coastal Eng.*, ASCE, Malaga, Spain, 492-504.

Mase, H. and Kirby, J. T., (1992). Hybrid frequency-domain KdV equation for random wave transformation. *Proc. of 23rd Int. Conf. Coastal Eng.*, ASCE, 474-487.

Mastenbroek, C., Burgers, G., and Janssen, P. A. E. M., (1993). The dynamical coupling of a wave model in a storm surge model through the atmospheric boundary layer. *J. Phys. Oceanogr.*, 23: 1856-1866.

Mei, C. C., (1983). The Applied Dynamics of Ocean Surface Waves. John Wiley and Sons Inc., New York, p. 740.

Miles, J. W., (1957). On the generation of surface waves by shear flows. *J. Fluid Mech.*, 3: 185-204.

Nelson, R. C., (1994). Depth limited wave heights in very flat regions. *Coastal Eng.*, 23: 43-59.

Nelson, R. C., (1987). Design Wave Heights on Very Mild Slopes: An Experimental Study. *Civil. Eng. Trans.,* 29: 157-161.

Padilla-Hernandez, R. and Monbaliu, J., (2001). Energy balance of wind waves as a function of the bottom friction formulation. *Coastal Eng.*, 43: 131-148.

Phillips, O. M., (1957). On the generation of waves by turbulent wind. *J. Fluid Mech.*, 2: 417-445.

Pierson, W. J. and Moskowitz, L., (1964). A proposed spectral form for fully developed wind seas based on the similarity theory of S.A. Kitaigorodskii. *J. Geophys. Res.*, 69(24): 5181-5190.

Plant, W. J., (1982). A relationship between stress and wave slope. *J. Geophys. Res.* 87(C#): 1961-1967.

Putnam, J. A. and Johnson, J. W., (1949). The dissipation of wave energy by bottom friction. *Trans. Am. Geoph. Union*, 30: 67-74.

Rogers, W. E., Kaihatu, J. M., Petit, H. A. H., Booij, N., and Holthuijsen, L. H., (2000). Arbitrary-Scale Propagation in a Third Generation Wind Wave Model, manuscript.

Seelig, W. N., (1979). Effects of breakwaters on waves: Laboratory tests of wave transmission by overtopping. *Proc. Conf. Coastal Structures*, 79(2): 941-961.

Shemdin, P., Hasselmann, K., Hsiao, S. V., and Herterich, K., (1978). Non-linear and linear bottom interaction effects in shallow water, in: Turbulent fluxes through the sea surface, *Wave Dynamics and Prediction, NATO Conf. Ser.*, V(1), 347-372 pp.

Snyder, R. L., Dobson, F. W., Elliott, J. A., and Long, R. B., (1981). Array measurement of atmospheric pressure fluctuations above surface gravity waves. *J. Fluid Mech.*, 102: 1-59.

Stelling, G. S. and Leendertse, J. J., (1992). Approximation of convective processes by cyclic AOI methods. *In the Proc. of the 2nd Int. Conf. on Estuarine and Coastal Modeling*, ASCE, Tampa, Florida, 771-782.

Thornton, E. B. and Guza, R. T., (1983). Transformation of wave height distribution. *J. Geophys. Res.*, 88(C10): 5925-5938.

Tolman, H. L., (1995). "On the selection of propagation schemes for a spectral wind-wave model." *NWS/NCEP Office Note 411*, 30 pp. + figures.

Tolman, H. L., (1992a). Effects of numerics on the physics in a third-generation wind-wave model. *J. Phys. Oceanogr.*, 22: 1095-1111.

Tolman, H. L., (1992b). An evaluation of expressions for the wave energy dissipation due to bottom friction in the presence of currents. *Coastal Eng.*, 16: 165-179.

Tolman, H. L., (1990). Wind wave propagation in tidal seas, *Ph.D. thesis*. Delft University of Technology, Department of Civil Engineering, the Netherlands.

Van der Vorst, H. A., (1992). Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for solution of non-symmetric linear systems. *SIAM J. Sci. Statistical Computing*, 13: 631-644.

Vincent, C. L., Smith, J. M., and Davis, J., (1994). Parameterization of wave breaking in models. M. Isaacson and M. Quick, eds., *Proc. of Int. Symp.: Waves - Physical and Numerical Modeling*, University of British Columbia, Vancouver, Canada, Vol. II, pp.753-762.

Vuik, C., (1993). Solution of the discretized incompressible Navier-Stokes equations with the GMRES method. *Int. J. for Num. Meth. in Fluids*, 16: 507-523.

WAMDI Group, (1988). The WAM model-A third generation ocean wave prediction model. *J. Phys. Oceanogr.*, 18: 1775-1810.

Weber, S. L., (1991a). Bottom friction for wind sea and swell in extreme depth-limited situations. *J. Phys. Oceanogr.*, 21: 149-172.

Weber, S. L., (1991b). Eddy-viscosity and drag-law models for random ocean wave dissipation. *J. Fluid Mech.*, 232: 73-98.

Weber, S. L., (1989). Surface gravity waves and turbulent bottom friction, *Ph.D. thesis*. University of Utrecht, the Netherlands.

Whitham, G. B., (1974). Linear and Nonlinear Waves. John Wiley and Sons Inc., New York, p. 636.

Wu, J., (1982). Wind-stress coefficients over sea surface from breeze to hurricane. *J. Geophys. Res.*, 87(C12): 9704-9706.

Yan, L., (1987). An improved wind input source term for third generation ocean wave Modeling. *Sci. Rep.*, WR 87-8, R. Neth. Meteorol. Inst., De Bilt, the Netherlands.

Young, I. R. and Van Vledder, G. P., (1993). A review of the central role of nonlinear interactions in wind-wave evolution. *Philos. Trans. R. Soc. London, Ser. A.*, 342: 505-524.

Young, I. R. and Banner, M. L., (1992). Numerical Experiments on the evolution of fetch limited waves, paper presented at *Int. Union of Theor. and Appl. Mech. (IUTAM)*, Sydney, Australia, 267-275.

Yuan, Yeli, Tung, C. C. and Huang N. E., (1986). "Statistical Characteristics of breaking waves" Wave Dynamics and Radio Probing of the Ocean Surface. O.M. Phillips and K. Hasselmann, eds., Plenum, New York, p. 265-272.

# 3.0 MODEL DESIGN DECISION

SWAN Version 40.01 has been modified to become Version 40.11. This section will discuss the additions, changes, compatibility, bug fixes and implementation of SWAN Version 40.11.

## 3.1 ADDITIONS TO SWAN

The first addition made to SWAN Version 40.11 allows for nesting in WAVEWATCH III. SWAN can now compute on spherical coordinates (latitude and longitude), allowing for calculations in laboratory situations, coastal regions, shelf seas and oceans. The new version also allows the user to define obstacles at which waves are reflected, such as coastlines or breakwaters, as opposed to just transmitting waves through obstacles. Lastly, a higher order propagation scheme was introduced for both the stationary and nonstationary modes.

## 3.2 CHANGES TO SWAN

The changes made to SWAN begin with the improvement of approximating the bathymetry in refraction computations. In order to give robust (but not necessarily accurate) results in cases of poor resolutions in bathymetry, currents or wave field, the user can now activate a limiter to avoid waves turning over more than 90 degrees in one spatial grid step. The limiter on the refraction is switched off on default. In Version 40.01 the Backward Space, Backward Time (BSBT) numerical propagation scheme was the only scheme available. Now, using Version 40.11 in stationary mode the Second ORDer UPwind (SORDUP) scheme is chosen as default, while in non-stationary mode the Stelling-scheme is default. The BSBT is still optionally available.

## 3.3 COMPATIBILITY OF SWAN

SWAN Version 40.11 is fully downward compatible with Version 40.01. Due to the changes in SWAN, a comparison of test results between Versions 40.01 and 40.11 may show differences in the results.

## 3.4 BUG FIXES

The purpose of describing the bug fixes, is to allow the user to identify previous SWAN runs that may have encountered these problems (either at runtime or in hindsight). The following are five bugs that were fixed in Version 40.11:

1. The output in the form of starplots on a rotated output frame;
2. The implementation of the QUANTITY command;
3. Spectral output of source terms on land points;
4. The output of 2-D spectra in combination with rotated grids or a directional sector;
5. The interpolation for test points too close to land points.

### 3.5 IMPLEMENTATION

SWAN (40.11) has been implemented so that all (except for one) obsolete FORTRAN 95 features have been removed to avoid compiler warnings. The implementation of allocatable arrays was done to avoid the use of the *pool* array for newly introduced arrays. Also implemented were modules to avoid lengthy argument lists of subroutines. The implementation of FORTRAN 90 implies that SWAN Version 40.11 will not compile under FORTRAN 77.

## 4.0 MODEL ARCHITECTURAL DESIGN

### 4.1 MODEL COMPONENTS

SWAN is a single computer program that is separated into three main files consisting of an executable file, a command file, and a run file.

**a.** Executable File- The name of the executable file is "a.out" for the versions running under Unix and "swanmain.exe" for the PC version generated with the Lahey Fortran90 compiler (in case swanmain.for is first on the list). Remove the "a.out" and replace with "swan.x" or "swan.exe".

**b.** Command File- The command file contains the user's input and selected instructions to run SWAN. The command file, which has the extension .swn must be presented to SWAN is American Standard Code for Information Interchange (ASCII) format.

**c.** Run File- Depending on which system is being used, either swan.bat (for MS-DOS systems) or swan.unix (for Unix systems) is the name of the run file. MS-DOS is not case-sensitive; however, Unix systems are.

## 4.2 SYSTEM REQUIREMENTS

The core memory for SWAN is determined at the installation of SWAN on the user's computer system. The required storage capacity in SWAN depends on the number of grid points in x- and y-direction (mxc*myc) and the number of points in frequency and directional space (msc*mdc). Calculating nonlinear four wave-wave interactions per sweep, instead of per iteration, decreases the amount of required memory by a factor of 2/3 (see Section 5.0 in the User's Manual-Carroll and Kelly, 2002). Other storage restrictions with calculating nonlinear four wave-wave interactions are summarized in Table 4.11-1 and 4.11-2 in the User's Manual (Carroll and Kelly, 2002).

To run the SWAN program for **test_cases**, 55 Mb of free internal memory is recommended. SWAN requires 100 to 500 Mb of memory for realistic cases, whereas for more stationary or 1-D cases significantly less memory is needed. The number of files addressable by the DOS system is at least twenty therefore the command line FILES=20 (or some higher number if necessary) should be included in the file config.sys of the DOS operating system.

## 4.3 CONCEPT OF EXECUTION

SWAN is a single program, consisting of an executable file with extension *.exe*, a command file with extension *.swn*, and a run file with either extension *.bat* or *.unix*, depending on whether or not MS-DOS or Unix is being used. The execution of SWAN consists of three steps 1) implementing SWAN on the user's computer, 2) editing the command file for a particular model run, and 3) running SWAN.

The first step, implementing SWAN, can be done in the following manner:

- Copy the source code and files from the SWAN web site (http://swan.ct.tudelft.nl/home.htm).
- Implement published bug fixes.
- Make the necessary modifications on dependent parts of code during installation.
- Compile the source code.
- Link the compiled source code.
- Test the executable SWAN and compare test results with those on the web site.

See the SWAN User's Manual for detailed information on implementation (Carroll and Kelly, 2002).

Next, the command file must be located and edited. The name of the command file from the source code will have the extension *.swn*. The user must present SWAN with one file (in ASCII) containing <u>all</u> of the actual commands. Within the command file the user should give the command's keyword, required or optional data, and comments. The keyword, which is usually the name of the command, indicates the primary function of that command and should comply with the rules of file identification of the computer system on which SWAN is run. To help with editing the input files for SWAN, the SWAN web site contains a template command file called *swan.edt*. The SWAN User's Manual provides a complete description of the commands available for selection in SWAN. Details of the command's keyword, data and comments and the way in which the user must enter them may be found in Appendix A of the User's Manual (Carroll and Kelly, 2002).

The final step is to run SWAN. Running SWAN requires three actions. First the user must copy the command file to INPUT (assuming INPUT is the standard filename for command input). Next, the user will run SWAN and view the output by copying the PRINT file (assuming PRINT is the standard filename for output). See **Section 4.4.1** for a description of the types of output files that are generated by SWAN.

A flow diagram illustrating the basic steps for the operation of SWAN is shown in **Figure 4.3-1**.

# Concept of Execution

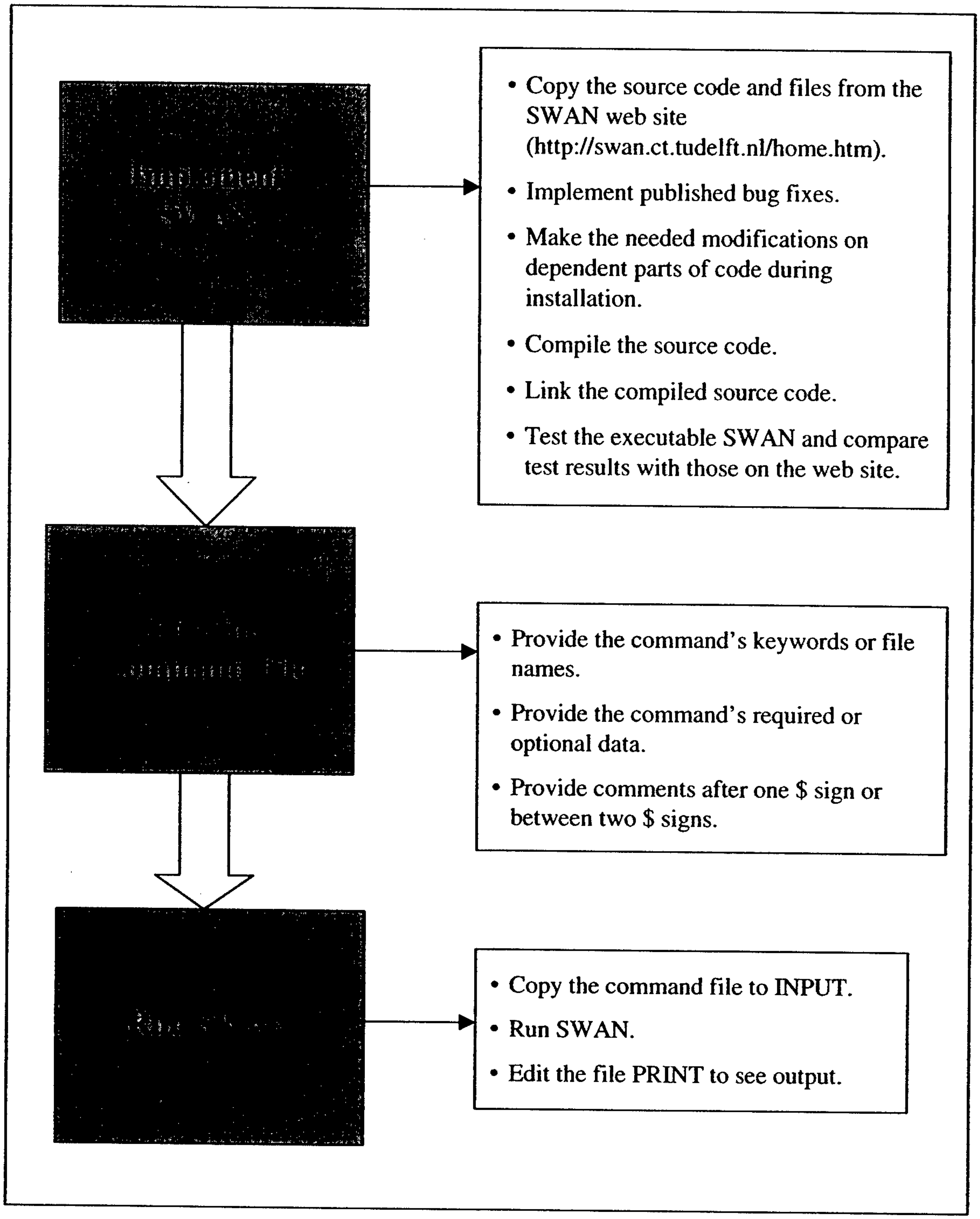


**Figure 4.3-1** Flow diagram summarizing the SWAN Version 40.11 execution steps.

## 4.4 INTERFACE DESIGN

### *4.4.1 Interface Identification and Diagrams*

The user must provide the following input files to SWAN:

- A command file containing the user selected instructions to run SWAN.
- File(s) containing the bottom current, friction, and wind (if relevant).
- File(s) containing the wave field at the model boundaries (if relevant).

SWAN produces output only at the user's request. The output is available for many different wave and wave-related parameters. The types of files generated by the output are given below:

- Print Files- Error messages appear in a PRINT file, which can be renamed by the user with a batch (DOS) or script (Unix) command. In the DOS and Unix systems the file PRINT is renamed to the name of the command file (examples are on the SWAN web site), with the extension *.swn* replaced by *.prt*. All files with extension *.prt* are referred to as print files.

- Numerical Output Files- Output from commands such as BLOCK or TABLE appears in files with user provided names.

- Plot Files- One or more plot files are generated by the PLOT command. If the user does not specify a filename the plot file has the name PLF... where the run number as defined in the command PROJECT as nr appears on the dots.

- Error Files- A file called ERRFILE, which contains the error messages, is created only when SWAN produces error messages. Existence of this file is an indication that results must be carefully examined.

- Grid Point Error Files- A file called ERRPTS contains the grid points where specific errors occurred during the calculation, such as non-convergence of the iterative matrix-solver. Existence of this file is an indication to study the grid point spectrum more carefully.

# 5.0 SWAN DETAILED DESIGN

## 5.1 CONSTRAINTS AND LIMITATIONS

Despite the improvements of Version 40.11, a few limitations still remain:

1. Diffraction is not modeled in SWAN, so SWAN should not be used in areas where variations in wave height are large within a horizontal scale of a few wavelengths. Because of this, the wave field computed by SWAN will generally not be accurate in the immediate vicinity of obstacles, and certainly not in harbors.

2. SWAN does not calculate wave-induced currents. If relevant, such currents should be provided as input to SWAN (e.g. from a hydrodynamical model, which can be driven by waves from SWAN in an iteration procedure). As an option SWAN computes wave induced set-up.

3. The Lumped Triad Approximation (LTA) used in triad wave-wave interactions seems to depend on the width of the directional distribution of the wave spectrum. The present tuning in SWAN works reasonably well in most cases. It was obtained from observations in a narrow wave flume (long-crested waves).

4. The Discrete Interaction Approximation (DIA) used in quadruplet wave-wave interactions depends on the width of the directional distribution of the wave spectrum. DIA works reasonably well in many cases but gives a poor approximation for long-crested waves (narrow directional distribution) that depend on the frequency resolution. The DIA has also proven to be a poor approximator of frequency resolutions very different from 10%. SWAN shares this fundamental limitation with other third-generation wave models such as WAM and WAVEWATCH III.

5. This version of SWAN (40.11) may be used on any scale relevant for wind generated surface gravity waves (high-quality propagation (third order diffusion) and Cartesian or spherical coordinates). The background for providing SWAN with such flexibility is to:

   - Allow SWAN to be used from laboratory conditions to shelf seas (but not harbors, see above) and
   - Nest SWAN in the WAM or WAVEWATCH III models, which are formulated in terms of spherical coordinates.

   These facilities are not meant to support the use of SWAN on oceanic scales. SWAN has not been extensively tested and is less efficient on oceanic scales than WAVEWATCH III and probably less efficient than WAM (SWAN does not parallelize or vectorize well). SWAN developers have no plans to apply SWAN to blue water.

There are a few constraints that the user might encounter:

1. Sometimes the user input to SWAN is such that SWAN produces unreliable and possibly even unrealistic results. This may be the case if the bathymetry or the wave field is not well resolved. Be aware that the grid on which the computations are performed interpolates from the grids on which the input is provided; different resolutions for these grids (which are allowed) can therefore create unexpected interpolation patterns on the computational grid.

2. Other problems are due to more fundamental shortcomings of SWAN (which may or may not be typical for third-generation wave models) and unintentional coding bugs such as:

   - The user can request that refraction over one spatial grid step is limited to 90°.
   - SWAN cannot handle wave propagation on super-critical current flow. If such flow is encountered during SWAN computations, the current is locally reduced to sub-critical flow.
   - If the water depth is less than some user-provided limit, the depth is set at that limit (default is 0.05 m).
   - SWAN may not reproduce the user-imposed wave boundary conditions as SWAN replaces the *imposed* waves that move out of the computational area at the boundaries with the *computed* waves that move out of the computational area at the boundaries.
   - SWAN may have convergence problems.

   Because of such scenarios, limiters, shortcomings and bugs, the results may look realistic but they may (locally) not be accurate. Any change in these limitations or problems (in particular newly discovered coding bugs and their fixes) are published on the SWAN web site (http://swan.ct.tudelft.nl) and implemented in new releases of SWAN.

## 5.2 LOGIC AND BASIC EQUATIONS

### *5.2.1 General Formulation*

The waves in SWAN are described with the two-dimensional wave action density spectrum, even when nonlinear phenomena dominate (e.g., in the surf zone). The rationale for using the spectrum in such highly nonlinear conditions is that even in these conditions it seems possible to predict with reasonable accuracy spectral distribution of the second order moment of the waves (although it may not be sufficient to fully describe the waves statistically). The spectrum that is considered in SWAN is the action density spectrum rather than the energy density spectrum since in the presence of currents, action density is conserved whereas energy density is not (Whitham, 1974). The independent variables are the relative frequency (as observed in a frame of reference moving with the action propagation velocity) and the wave direction (the direction normal to the wave crest of each spectral component). The action density is equal to the energy density divided by the relative frequency. In SWAN, this spectrum may vary in time and space.

#### 5.2.1.1 Action Balance Equation

The evolution of the wave spectrum in SWAN is described by the spectral action balance equation, which for Cartesian coordinates is (e.g., Hasselmann et al., 1973):

$$\frac{\partial}{\partial t}N+\frac{\partial}{\partial x}c_xN+\frac{\partial}{\partial y}c_yN+\frac{\partial}{\partial \sigma}c_\sigma N+\frac{\partial}{\partial \theta}c_\theta N=\frac{S}{\sigma} \tag{1a}$$

The first term in the left-hand side of this equation represents the local rate of change of action density in time, the second and third term represent propagation of action in geographical space (with propagation velocities $c_x$ and $c_y$ in $x$- and $y$-space, respectively). The fourth term represents shifting of the relative frequency due to variations in depths and currents (with propagation velocity $c_\sigma$ in $\sigma$-space). The fifth term represents depth-induced and current-induced refraction (with propagation velocity $c_\theta$ in $\theta$ -space). The expressions for these propagation speeds are taken from linear wave theory (e.g., Whitham, 1974; Mei, 1983; and Dingemans, 1997). The term $S$ (= $S(\sigma,\theta)$) at the right hand side of the action balance equation is the source term in terms of energy density representing the effects of generation, dissipation and nonlinear wave-wave interactions. A brief summary of the formulations that are used for the various source terms in SWAN is given next.

In view of the use of SWAN at shelf, sea or oceanic scales the user can choose to express the basic equation in spherical coordinates:

$$\frac{\partial}{\partial t}N+\frac{\partial}{\partial \lambda}c_\lambda N+(\cos\varphi)^{-1}\frac{\partial}{\partial \varphi}c_\varphi \cos\varphi N+\frac{\partial}{\partial \sigma}c_\sigma N+\frac{\partial}{\partial \theta}c_\theta N=\frac{S}{\sigma} \tag{1b}$$

with longitude, $\lambda$ and latitude, $\varphi$.

#### 5.2.1.2 Wind Input

Transfer of wind energy to the waves is described in SWAN with a resonance mechanism (Phillips, 1957) and a feedback mechanism (Miles, 1957). The corresponding source term for these mechanisms is commonly described as the sum of linear and exponential growth:

$$S_{in}(\sigma,\theta)=A+BE(\sigma,\theta) \tag{2}$$

in which $A$ and $B$ depend on wave frequency and direction, and wind speed and direction. The effects of currents are accounted for in SWAN by using the apparent local wind speed and direction. The expression for term $A$ is due to Cavaleri and Malanotte-Rizzoli (1981) with a filter to avoid growth at frequencies lower than the Pierson-Moskowitz frequency (Tolman, 1992a). Two optional expressions for coefficient $B$ are used in the model. The first is taken from an early version of the WAM model (known as WAM Cycle 3, the WAMDI group, 1988). This is due to Snyder et al. (1981), rescaled in terms of friction velocity $U$ by Komen et al. (1984). The drag coefficient to relate $U$ to the driving wind speed at 10m elevation $U_{10}$ is taken from Wu (1982). The second expression for $B$ is taken from the most recent version of the WAM model (known as WAM Cycle 4, Komen et al., 1994). It is due to Janssen (1991a) and accounts explicitly for the interaction between the wind and the waves by considering atmospheric boundary layer effects and the roughness length of the sea surface. The corresponding set of equations is solved (as in the WAM model) with the iterative procedure of Mastenbroek et al. (1993).

#### 5.2.1.3 Dissipation

The dissipation term of wave energy is represented by the summation of three different contributions: whitecapping, $s_{ds,w}(\sigma, \theta)$, bottom friction, $s_{ds,b}(\sigma, \theta)$, and depth-induced breaking, $s_{ds,br}(\sigma, \theta)$.

Whitecapping is primarily controlled by the steepness of the waves. In presently operating third-generation wave models (including SWAN) the whitecapping formulations are based on a pulse-based model (Hasselmann, 1974) as adapted by the WAMDI group (1988):

$$s_{ds,w}(\sigma,\theta)=-\Gamma\tilde{\sigma}\frac{k}{\tilde{k}}E(\sigma,\theta) \tag{3}$$

where $\Gamma$ is a steepness dependent coefficient, $k$ is wave number and $\tilde{\sigma}$ and $\tilde{k}$ denotes a mean frequency and a mean wave number, respectively (cf. the WAMDI group, 1988). Komen et al. (1984) estimated the value of $\Gamma$ by closing the energy balance of the waves in fully developed conditions. This implies that this value depends on the wind-input formulation that is used. Since two expressions are used for the wind input in SWAN, two values for $\Gamma$ are also used. The first is due to Komen et al. (1984), as in Cycle 3 of the WAM model. It is used in SWAN when the wind input coefficient $B$ of Komen et al. (1984) is used. The second expression is an adaptation of this expression based on Janssen (1991a); as in Cycle 4 of the WAM model (see Janssen, 1991b and Günther et al., 1992). It is used when the wind input term $B$ of Janssen (1991a) is used. Young and Banner (1992) and Banner and Young (1994) have shown that the results of closing the energy balance in this manner depend critically on the choice of a high-frequency cut-off frequency above which a diagnostic spectral tail is used. In SWAN, this cut-off frequency is different from the one used in the WAM model. Differences in the growth rates between the WAM model and SWAN are therefore to be expected.

Depth-induced dissipation may be caused by bottom friction, by bottom motion, by percolation or by back scattering on bottom irregularities (Shemdin et al., 1978). For continental shelf seas with sandy bottoms, the dominant mechanism appears to be bottom friction (e.g., Bertotti and Cavaleri, 1994) which can be represented as:

$$S_{ds,b}(\sigma,\theta) = -C_{bottom} \frac{\sigma^2}{g^2 \sinh^2(kd)} E(\sigma,\theta) \tag{4}$$

in which $C_{bottom}$ is a bottom friction coefficient. A large number of models have been proposed since the pioneering paper of Putnam and Johnson (1949). Hasselmann et al., (1973) suggested using an empirically obtained constant. It seems to perform well in many different conditions as long as a suitable value is chosen (typically different for swell and wind sea; (Bouws and Komen, 1983)). Hasselmann and Collins (1968) which was later simplified by Collins (1972) have proposed a nonlinear formulation based on drag. More complicated, eddy viscosity models have been developed by Madsen et al. (1988) and by Weber (1989, 1991a, 1991b). Considering the large variations in bottom conditions in coastal areas (bottom material, bottom roughness length, ripple height etc.), there is no field data evidence to give preference to a particular friction model (Luo and Monbaliu, 1994). For this reason, the simplest of each of these types of friction models has been implemented in SWAN: the empirical JONSWAP model of Hasselmann et al. (1973), the drag law model of Collins (1972) and the eddy-viscosity model of Madsen et al. (1988). The effect of a mean current on the wave energy dissipation due to bottom friction is not taken into account in SWAN. The reasons for this are given by Tolman (1992b) who argues that state-of-the-art expressions vary too widely in their effects to be acceptable. He found that the error in finding a correct estimate of the bottom roughness length scale has a much larger impact on the energy dissipation rate than the effect of a mean current.

The process of depth-induced wave breaking is still poorly understood and little is known about its spectral modeling. In contrast to this, the total dissipation (i.e., integrated over the spectrum) due to this type of wave breaking can be well modeled with the dissipation of a bore applied to the breaking waves in a random field (Battjes and Janssen, 1978 and Thornton and Guza, 1983). Laboratory observations (Battjes and Beji, 1992, Vincent et al. 1994; Arcilla et al., 1994 and Eldeberky and Battjes, 1996) show that the shape of initially uni-modal spectra propagating across simple (barred) beach profiles, is fairly insensitive to depth-induced breaking. This has led Eldeberky and Battjes (1995) to formulate a spectral version of the bore model of Battjes and Janssen (1978) which conserves the spectral shape. Expanding their expression to include directions, the expression that is used in SWAN is:

$$S_{ds,br}(\sigma,\theta)=\frac{D_{tot}}{E_{tot}}E(\sigma,\theta) \tag{5}$$

in which $E_{tot}$ is the total wave energy and $D_{tot}$(which is negative) is the rate of dissipation of the total energy due to wave breaking according to Battjes and Janssen (1978). Adding a quadratic dependency on frequency as suggested by Mase and Kirby (1992) supported by Elgar et al. (1997) seems to have no noticeable effect on the SWAN results. Chen and Guza (1997) inferred from observations and simulations with a Boussinesq model that the high-frequency levels are insensitive to such frequency dependency because an increased dissipation at high frequencies is compensated approximately by increased nonlinear energy transfer (but they did find the frequency dependency to be relevant in time domain). The value of $D_{tot}$ depends critically on the breaking parameter $\gamma = H_{max}/d$ (in which $H_{max}$ is the maximum possible individual wave height in the local water depth). In SWAN, a constant value and a variable value are available. The constant value is $\gamma =$ 0.73 (the mean value of the data set of Battjes and Stive (1985)).

SWAN can estimate wave transmission through a (line-) structure such as a breakwater (dam). Such an obstacle will affect the wave field in two ways, first it will reduce the wave height locally all along its length, and second it will cause diffraction (which the model does not account for) around its end(s). In irregular, short-crested wave fields, however; it seems that the effect of diffraction is small, except in a region less than one or two wavelengths away from the tip of the obstacle (Booij et al., 1993). Therefore the model can reasonably account for waves around an obstacle if the directional spectrum of incoming waves is not too narrow. Since obstacles usually have a transversal area that is too small to be resolved by the bottom grid in SWAN, an obstacle is modeled as a line. If the crest of the breakwater is at a level where (at least part of the) waves can pass over, the transmission coefficient $K_t$ (defined as the ratio of the (significant) wave height at the downwave side of the dam over the (significant) wave height at the upwave side) is a function of wave height and the difference in crest level and water level. The expression is taken from Goda et al. (1967):

$$K_t = 0.5\left[1-\sin\left(\frac{\pi}{2\alpha}\left(\frac{F}{H_i}+\beta\right)\right)\right] \quad \text{for } -\beta-\alpha<\frac{F}{H_i}<\alpha-\beta \tag{6}$$

where $F = h$-$d$ is the freeboard of the dam and where $H_i$ is the incident (significant) wave height at the upwave side of the obstacle (dam), $h$ is the crest level of the dam above the reference level same as reference level of the bottom), $d$ the mean water level relative to the reference level, and the coefficients $\alpha, \beta$ depend on the shape of the dam. Table 5.2-1 provides the coefficients for some of the more common cases encountered.

**Table 5.2-1.** Coefficients $\alpha$, $\beta$ determined by the shape of the dam (Seelig, 1979).

| Case | $\alpha$ | $\beta$ |
|---|---|---|
| vertical thin wall | 1.8 | 0.1 |
| caisson | 2.2 | 0.4 |
| dam with slope 1:3/2 | 2.6 | 0.15 |

Equation 6 is based on experiments in a wave flume, so strictly speaking it is only valid for normal incidence waves. Since there are no data available on oblique waves it is assumed that the transmission coefficient does not depend on direction. Another phenomenon that is to be expected is a change in wave frequency since often the process above the dam is highly nonlinear. Again there is little information available, so in SWAN it is assumed that the frequencies remain unchanged over an obstacle (only the energy scale of the spectrum is affected and not the spectral shape).

#### 5.2.1.4 Nonlinear Wave-wave Interactions

In deep water, quadruplet wave-wave interactions dominate the evolution of the spectrum. These interactions transfer wave energy from the spectral peak to lower frequencies (thus moving the peak frequency to lower values) and to higher frequencies (where the energy is dissipated by whitecapping). In very shallow water, triad wave-wave interactions transfer energy from lower to higher frequencies often resulting in higher harmonics (Beji and Battjes, 1993); low-frequency energy generation by triad wave-wave interactions is not considered here.

A full computation of the quadruplet wave-wave interactions is extremely time consuming and not convenient in any operational wave model. A number of techniques, based on parametric methods or other types of approximations have been proposed to improve computational speed (see Young and Van Vledder, 1993 for a review). In SWAN the computations are carried out with the DIA of Hasselmann et al. (1985). This DIA has been found quite successful in describing the essential features of a developing wave spectrum (Komen et al., 1994). For uni-directional waves, this approximation is not valid. In fact, the quadruplet interaction coefficient for these waves is nearly zero (G. P. van Vledder, personal communication, 1996). For finite-depth applications, Hasselmann and Hasselmann (1981) have shown that for a JONSWAP-type spectrum the quadruplet wave-wave interactions can be scaled with a simple expression (it is used in SWAN).

A first attempt to describe triad wave-wave interactions in terms of a spectral energy source term was made by Abreu et al. (1992). However, their expression is restricted to non-dispersive shallow water waves and is therefore not suitable in many practical applications of wind waves. The breakthrough in the development came with the work of Eldeberky and Battjes (1995), which transformed the amplitude part of the Boussinesq model of Madsen and Sørensen (1993) into an energy density formulation and parameterized the biphase of the waves on the basis of laboratory observations (Battjes and Beji, 1992 and Arcilla et al., 1994). A Discrete Triad Approximation (DTA) for co-linear waves was subsequently obtained by considering only the dominant self-self interactions. The Boussinesq model has been verified with flume observations of long-crested, random waves breaking over a submerged bar (Beji and Battjes, 1993) and over a barred beach (Arcilla et al., 1994). The model appeared to be fairly successful in describing the essential features of the energy transfer from the primary peak of the spectrum to the super harmonics. The LTA, a slightly different version derived by Eldeberky (1996) is used in SWAN.

### *5.2.2 First-, Second- and Third-generation Mode*

SWAN can operate in first-, second- and third-generation mode. The first- and second-generation modes are essentially those of Holthuijsen and De Boer (1988) as indicated above (first-generation with a constant Phillips constant of 0.0081; second-generation with a variable Phillips constant). An overview of the options is given in **Table 5.2-2**.

**Table 5.2-2:** Summary of options available for SWAN operation modes.

| Option | Source | Generation mode of SWAN | | |
|---|---|---|---|---|
| | | 1st | 2nd | 3rd |
| Linear wind growth: | Cavaleri and Malanotte-Rizzoli (1981) [modified] | x | x | |
| | Cavaleri and Malanotte-Rizzoli (1981) | | | x |
| Exponential wind growth: | Snyder et al. (1981) [modified] | x | x | |
| | Snyder et al. (1981) | | | x[1] |
| | Janssen (1989, 1991) | | | x[2] |
| Whitecapping: | Holthuijsen and De Boer (1988) | x[3] | x[4] | |
| | Komen et al. (1984) | | | x[1] |
| | Janssen (1991), Komen et al. (1994) | | | x[2] |
| Quadruplet interaction: | Hasselmann et al. (1985) | | | x |
| Triad interactions: | Eldeberky (1996) | x | x | x |
| Depth-induced breaking: | Battjes and Janssen (1978) | x | x | x |
| Bottom friction: | Hasselmann et al. (1973) | x | x | x |
| | Collins (1972) | x | x | x |
| | Madsen et al. (1988) | x | x | x |
| Obstacle transmission: | Seelig (1979) | x | x | x |

For SWAN running in a third generation mode, the following combinations of the input and whitecapping parameterizations are used (indicated with 1 and 2, see command GEN3):

1. Gives the wind input and whitecapping formulations as used in WAM Cycle 3.
2. Gives the wind input and whitecapping formulations as used in WAM Cycle 4.
3. Pierson-Moskowitz spectrum as an upper limit.

4. Scaled Pierson-Moskowitz spectrum as upper limit.

### *5.2.3 Wave-induced Set-up*

In a (geographic) 1-D case the computation of the wave-induced set-up is based on the vertically integrated momentum balance equation which reduces to a balance between the gradient of the wave radiation stress and the hydrodynamic pressure gradient (no wave-induced currents exist). In a 2-D case the computation of the wave-induced set-up is based on the divergence of the vertically integrated momentum balance equation equaling zero.

### *5.2.4 Detailed Formulation*

The complete expressions for the physical processes of generation, dissipation and nonlinear wave-wave interactions that are available in the SWAN model are given here.

#### 5.2.4.1 Input by Wind ($S_{in}$)

Wave growth by wind is described by:

$$S_{in}(\sigma,\theta) = A + BE(\sigma,\theta) \tag{7}$$

in which $A$ describes linear growth and $BE$ exponential growth. It should be noted that the SWAN model is driven by the wind speed at 10m elevation $U_{10}$ whereas the computations use the friction velocity $U_*$. For the WAM Cycle 3 formulation the transformation from $U_{10}$ to $U_*$ is obtained with

$$U_*^2 = C_D U_{10}^2 \quad , \tag{8}$$

in which $C_D$ is the drag coefficient from Wu (1982):

$$C_D(U_{10}) = \begin{cases} 1.2875\times10^{-3} & \text{for}\, U_{10} < 7.5\,\text{m/s} \\ (0.8 + 0.065\,\text{s/m}\times U_{10})\times10^{-3} & \text{for}\, U_{10} \geq 7.5\,\text{m/s} \end{cases} \tag{9}$$

For the WAM Cycle 4 formulations, the computation of $U_*$ is an integral part of the source term.

#### 5.2.4.2 Linear Growth by Wind

For the linear growth term $A$, the expression due to Cavaleri and Malanotte-Rizzoli (1981) is used with a filter to eliminate wave growth at frequencies lower than the

Pierson-Moskowitz frequency (Tolman, 1992a) (note that in his Eq. 10 the power of $10^{-5}$ should be $10^{-3}$, H. Tolman, personal communication, 1995):

$$A = \frac{1{,}5 \times 10^{-3}}{g^2 2\pi} [U_* \max[0, \cos(\theta - \theta_w)]]^4 H \quad ,$$

$$H = \exp(-(\sigma / \sigma^*_{PM})^{-4}) \quad \text{with} \quad \sigma^*_{PM} = \frac{0.13 g}{28 U_*} 2\pi \quad , \tag{10}$$

in which $\theta_w$ is the wind direction, $H$ is the filter and $\theta_{PM}{}^*$ is the peak frequency of the fully developed sea state according to Pierson and Moskowitz (1964; reformulated in terms of friction velocity).

#### 5.2.4.3 Exponential Growth by Wind

Two expressions for exponential growth by wind are optionally available in the SWAN model. The first expression is due to Komen et al. (1984). Their expression is a function of $\frac{U_*}{C_{ph}}$:

$$B = \max\left[0, 0.25 \frac{\rho_a}{\rho_w}\left[28 \frac{U_*}{C_{ph}} \cos(\theta - \theta_w) - 1\right]\right]\sigma \quad , \tag{11}$$

in which $c_{ph}$ is the phase speed and $\rho_a$ and $\rho_w$ are the density of air and water, respectively. This expression is also used in WAM Cycle 3 (WAMDI group, 1988). The second expression, which is based on a quasi-linear wind-wave theory, is due to Janssen (1989, 1991) and is given by:

$$B = \beta \frac{\rho_a}{\rho_w} \left(\frac{U_*}{c_{ph}}\right)^2 \max[0, \cos(\theta - \theta_w)]^2 \sigma \tag{12}$$

where $\beta$ is the Miles constant. In the theory of Janssen (1991), this Miles constant is estimated from the non-dimensional critical height $\lambda$:

$$\begin{cases} \beta = \dfrac{1.2}{\kappa^2} \lambda \ln^4 \lambda \ , & \lambda \le 1 \\ \lambda = \dfrac{g\, z_e}{c^2_{ph}} e^r , & r = \kappa c / \left| U_* \cos(\theta - \theta_w) \right| \end{cases} \tag{13}$$

where $\kappa$ is the Von Karman constant, equal to 0.41 and $z_e$ is the effective surface roughness. If the non-dimensional critical height $\lambda > 1$, the Miles constant $\beta$ is set equal to zero. Janssen (1991) assumes that the wind profile is given by:

$$U(z) = \frac{U_*}{\kappa} \ln\left( \frac{z + z_e - z_o}{z_e} \right) \quad , \tag{14}$$

in which $U(z)$ is the wind speed at height $z$ (10m in the SWAN model) above the mean water level, $z_o$ is the roughness length. The effective roughness length $z_e$ depends on the roughness length $z_o$ and the sea state through the wave induced stress $\tau_w$ and the total surface stress $\tau$:

$$z_e = \frac{z_o}{\sqrt{1 - \tau_w / \tau}} \quad \text{and} \quad z_o = \hat{\alpha} \frac{U_*^2}{g} \quad , \tag{15}$$

The second of these two equations is a Charnock-like relation in which $\hat{\alpha}$ is a constant equal to 0.01. The wave stress $\underline{\tau}_w$ vector is given by:

$$\underline{\tau}_w = \rho_w \int_0^{2\pi}\int_0^{\infty} \sigma\, B\, E(\sigma, \theta) \frac{\underline{k}}{k} \mathrm{d}\sigma \mathrm{d}\theta \quad . \tag{16}$$

The value of $U_*$ can be determined for a given wind speed $U_{10}$ and a given wave spectrum $E(\sigma, \theta)$ from the above set of equations. In the SWAN model the iterative procedure of Mastenbroek et al. (1993) is used. This set of expressions (Eq. 12 - 16) is also used in WAM Cycle 4 (Komen et al., 1994).

#### 5.2.4.4 Dissipation of Wave Energy ($S_{ds}$)

##### *5.2.4.4.1 Whitecapping*

The pulse-based model of Hasselmann (1974) represents the processes of whitecapping in the SWAN model. Reformulated in terms of wave number (rather than frequency) so as to be applicable in finite water depth (cf. WAMDI group, 1988), this expression is:

$$S_{ds,w}(\sigma, \theta) = -\Gamma \tilde{\sigma} \frac{k}{\tilde{k}} E(\sigma, \theta) \quad , \tag{17}$$

where $\tilde{\sigma}$ and $\tilde{k}$ denote the mean frequency and the mean wave number (for expressions see below) respectively, and the coefficient $\Gamma$ depends on the overall wave steepness. This steepness dependent coefficient, as given by the WAMDI group (1988), has been adapted by Günther et al. (1992) based on Janssen (1991a, 1991b):

$$\Gamma = \Gamma_{KJ} = C_{ds}\left((1-\delta)+\delta\frac{k}{\tilde{k}}\right)\left(\frac{\tilde{s}}{\tilde{s}_{PM}}\right)^{p} \quad . \tag{18}$$

For $\delta = 0$ the expression of $\Gamma$ reduces to the expression as used by the WAMDI group (1988). The coefficients $C_{ds}$, $\delta$ and m are tunable coefficients, $\tilde{s}$ is the overall wave steepness (defined below), $\tilde{s}_{PM}$ is the value of $\tilde{s}$ for the Pierson-Moskowitz spectrum (1964; $\tilde{s}_{PM} = (3.02 \times 10^{-3})^{1/2}$). This overall wave steepness $\tilde{s}$ is defined as:

$$\tilde{s} = \tilde{k}\sqrt{E_{tot}} \quad . \tag{19}$$

The mean frequency $\tilde{\sigma}$, the mean wave number $\tilde{k}$, and the total wave energy $E_{tot}$ are defined as (cf. WAMDI group, 1988):

$$\tilde{\sigma} = \left(E_{tot}^{-1}\int_0^{2\pi}\int_0^{\infty}\frac{1}{\sigma}E(\sigma,\theta)\,d\sigma\,d\theta\right)^{-1} ,$$
$$\tilde{k} = \left(E_{tot}^{-1}\int_0^{2\pi}\int_0^{\infty}\frac{1}{\sqrt{k}}E(\sigma,\theta)\,d\sigma\,d\theta\right)^{-2} \tag{20}$$

$$E_{tot} = \int_0^{2\pi}\int_0^{\infty}E(\sigma,\theta)\,d\sigma\,d\theta \quad . \tag{21}$$

The values of the tunable coefficients $C_{ds}$ and $\delta$ and exponent $p$ in this model have been obtained by Komen et al., (1984) and Janssen (1992) by closing the energy balance of the waves in idealized wave growth conditions (both for growing and fully developed wind seas) for deep water. This implies that coefficients in the steepness dependent coefficient $\Gamma$ depend on the wind-input formulation that is used. Since two different wind input formulations are used in the SWAN model, two sets of coefficients are used. For the wind input of Komen et al. (1984; corresponding to WAM Cycle 3; the WAMDI group, 1988): $C_{ds} = 2.36\times10^{-5}$, $\delta = 0$ and $p = 4$. Janssen (1992) and Günther (1992) obtained (assuming $p = 4$) $C_{ds} = 4.10\times10^{-5}$ and $\delta = 0.5$ (as used in the WAM Cycle 4; Komen et al., 1994).

#### *5.2.4.4.2 Bottom Friction*

The bottom friction models that have been selected for SWAN are the empirical model of JONSWAP (Hasselmann et al., 1973), the drag law model of Collins (1972) and the eddy-viscosity model of Madsen et al. (1988). The formulations for these bottom friction models can all be expressed in the following form:

$$S_{ds,b}(\sigma,\theta) = -C_{bottom}\frac{\sigma^2}{g^2\sinh^2(kd)}E(\sigma,\theta) \quad , \tag{22}$$

in which $C_{bottom}$ is a bottom friction coefficient that generally depends on the bottom orbital motion represented by $U_{rms}$:

$$U_{rms}^2 = \int_0^{2\pi}\int_0^{\infty}\frac{\sigma^2}{\sinh^2(kd)}E(\sigma,\theta)\,d\sigma\,d\theta \quad . \tag{23}$$

Hasselmann et al. (1973) found from the results of the JONSWAP experiment $C_{bottom} = C_{JON} = 0.038\ m^2s^{-3}$ for swell conditions. Bouws and Komen (1983) selected a bottom friction coefficient of $C_{JON} = 0.067\ m^2s^{-3}$ for fully developed wave conditions in shallow water. Both values are available in SWAN.

The expression of Collins (1972) is based on a conventional formulation for periodic waves with the appropriate parameters adapted to suit a random wave field. The dissipation rate is calculated with the conventional bottom friction formulation of Eq. 7 in which the bottom friction coefficient is $C_{bottom} = C_f g U_{rms}$ with $C_f = 0.015$ (Collins, 1972). (Note that Collins (1972) contains an error in the expression due to an erroneous Jacobean transformation; see page A-16 of Tolman, 1990).

Madsen et al. (1988) derived a formulation similar to that of Hasselmann and Collins (1968), but in their model the bottom friction factor is a function of the bottom roughness height and the actual wave conditions. Their bottom friction coefficient is given by:

$$C_{bottom} = f_w\frac{g}{\sqrt{2}}U_{rms} \quad , \tag{24}$$

in which $f_w$ is a non-dimensional friction factor estimated by using the formulation of Jonsson (1966; cf. Madsen et al., 1988):

$$\frac{1}{4\sqrt{f_w}} + \log_{10}\left[\frac{1}{4\sqrt{f_w}}\right] = m_f + \log_{10}\left[\frac{a_b}{K_N}\right] \quad , \tag{25}$$

in which $m_f = -0.08$ (Jonsson and Carlsen, 1976) and $a_b$ is a representative near-bottom excursion amplitude:

$$a_b^2 = 2\int_0^{2\pi}\int_0^{\infty}\frac{1}{\sinh^2(kd)}E(\sigma,\theta)\,d\sigma\,d\theta \quad , \tag{26}$$

and $K_N$ is the bottom roughness length scale. For values of $a_b / K_N$ smaller than 1.57 the friction factor $f_w$ is 0.30 (Jonsson, 1980).

#### *5.2.4.4.3 Depth-induced Wave Breaking*

To model the energy dissipation in random waves due to depth-induced breaking, the bore-based model of Battjes and Janssen (1978) is used in SWAN. The mean rate of energy dissipation per unit horizontal area due to wave breaking $D_{tot}$ is expressed as:

$$D_{tot} = -\frac{1}{4}\alpha_{BJ} Q_b \left(\frac{\bar{\sigma}}{2\pi}\right) H_m^2 \quad , \tag{27}$$

in which $\alpha_{BJ} = 1$ in SWAN, $Q_b$ is the fraction of breaking waves determined by:

$$\frac{1-Q_b}{\ln Q_b} = -8\frac{E_{tot}}{H_m^2} \quad , \tag{28}$$

in which $H_m$ is the maximum wave height that can exist at the given depth and $\bar{\sigma}$ is a mean frequency defined as:

$$\bar{\sigma} = E_{tot}^{-1} \int_0^{2\pi}\int_0^{\infty} \sigma E(\sigma,\theta)\,d\sigma\, d\theta \quad . \tag{29}$$

Extending the expression of Eldeberky and Battjes (1995) to include the spectral directions, the dissipation for a spectral component per unit time is calculated in SWAN with:

$$S_{ds,br}(\sigma,\theta) = D_{tot}\frac{E(\sigma,\theta)}{E_{tot}} \quad , \tag{30}$$

The maximum wave height, $H_m$, is determined in SWAN with $H_m = \gamma d$, in which $\gamma$ is the breaker parameter and $d$ is the total water depth (including the wave-induced set-up if computed by SWAN). In the literature, this breaker parameter $\gamma$ is often a constant or it is expressed as a function of bottom slope or incident wave steepness (see e.g., Galvin, 1972; Battjes and Janssen, 1978; Battjes and Stive, 1985; Arcilla and Lemos, 1990; Kaminsky and Kraus, 1993; and Nelson, 1987, 1994). Since SWAN is locally defined, the dependency on incident wave steepness cannot be used. Instead, the other two options (constant value or bottom-slope dependent) were used in SWAN Version 40.01 and older to determine the value of the breaker parameter. In SWAN III Version 40.11 the option of Nelson has been removed as the results of SWAN were better with the option of a constant value.

In the publication of Battjes and Janssen (1978) in which the dissipation model is described, a constant breaker parameter of $\gamma = 0.8$ was used based on Miche's criterion. Battjes and Stive (1985) re-analyzed wave data of a number of laboratory and field experiments and found values for the breaker parameter varying between 0.6 and 0.83 for different types of bathymetry (plane, bar-trough and bar) with an average of 0.73. From a compilation of a large number of experiments Kaminsky and Kraus (1993) have found breaker parameters in the range of 0.6 to 1.59 with an average of 0.79.

#### 5.2.4.5 Nonlinear Wave-wave Interactions ($S_{nl}$)

##### *5.2.4.5.1 Quadruplet Wave-wave Interactions*

The quadruplet wave-wave interactions are computed with the DIA as proposed by Hasselmann et al. (1985). Their source code (slightly adapted by Tolman, personal communication, 1993) has been used in the SWAN model. In the DIA two quadruplets of wave numbers are considered, both with frequencies:

$$\begin{aligned} \sigma_1 &= \sigma_2 = \sigma \\ \sigma_3 &= \sigma(1+\lambda) = \sigma^+ \quad , \\ \sigma_4 &= \sigma(1-\lambda) = \sigma^- \end{aligned} \tag{31}$$

where $\lambda$ is a constant coefficient set equal to 0.25. To satisfy the resonance conditions for the first quadruplet, the wave number vectors with frequency $\sigma_3$ and $\sigma_4$ lie at an angle of $\theta_1 = -11.5°$ and $\theta_2 = 33.6°$ to the two identical wave number vectors with frequencies $\sigma_1$ and $\sigma_2$. The second quadruplet is the mirror of this first quadruplet (the wave number vectors with frequency $\sigma_3$ and $\sigma_4$ lie at mirror angles of $\theta_3 = 11.5°$ and $\theta_4 = -33.6°$).

Within this discrete interaction approximation, the source term $S_{nl4}(\sigma, \theta)$ is given by:

$$S_{nl4}(\sigma,\theta) = S^{*}_{nl4}(\sigma,\theta) + S^{**}_{nl4}(\sigma,\theta) \quad , \tag{32}$$

where $S^{*}_{nl4}$ refers to the first quadruplet and $S^{**}_{nl4}$ to the second quadruplet (the expressions for $S^{**}_{nl4}$ are identical to those for $S^{*}_{nl4}$ for the mirror directions) and:

$$S^{*}_{nl4}(\sigma,\theta) = 2\delta S_{nl4}(\alpha_1\sigma,\theta) - \delta S_{nl4}(\alpha_2\sigma,\theta) - \delta S_{nl4}(\alpha_3\sigma,\theta) \quad , \tag{33}$$

in which $\alpha_1 = 1$, $\alpha_2 = (1 + \lambda)$ and $\alpha_3 = (1 - \lambda)$. Each of the contributions ($i = 1, 2, 3$) is:

$$\delta S_{nl4}(\alpha_i\sigma,\theta) = C_{nl4}(2\pi)^2 g^{-4}\left(\frac{\sigma}{2\pi}\right)^{11}\left[E^2(\alpha_i\sigma,\theta)\left(\frac{E(\alpha_i\sigma^+,\theta)}{(1+\lambda)^4}+\frac{E(\alpha_i\sigma^-,\theta)}{(1-\lambda)^4}\right) - 2\frac{E(\alpha_i\sigma,\theta)E(\alpha_i\sigma^+,\theta)E(\alpha_i\sigma^-,\theta)}{(1-\lambda^2)^4}\right] . \tag{34}$$

The constant $C_{nl4} = 3\times10^7$ . Following Hasselmann and Hasselmann (1981), the quadruplet interaction in finite water depth is taken identical to the quadruplet transfer in deep water multiplied with a scaling factor $R$:

$$S_{nl4,\,finitedepth} = R(k_p d)\, S_{nl4,infinitedepth} \quad , \tag{35}$$

where $R$ is given by:

$$R(k_p d) = 1 + \frac{C_{sh1}}{k_p d}\left(1 - C_{sh2}\cdot k_p d\right)\exp\left(C_{sh3}\cdot k_p d\right) \quad , \tag{36}$$

in which $k_p$ is the peak wave number of the JONSWAP spectrum for which the original computations were carried out. The values of the coefficients are: $C_{sh1} = 5.5$, $C_{sh2} = 6/7$ and $C_{sh3} = -1.25$. In the shallow water limit, i.e., $k_p d \to 0$ the nonlinear transfer tends to infinity. Therefore a lower limit of $k_p d = 0.5$ is applied (cf. WAM Cycle 4; Komen et al., 1994), resulting in a maximum value of $R(k_p d) = 4.43$. To increase the model robustness in case of arbitrarily shaped spectra, the peak wave number $k_p$ is replaced by $k_p = 0.75\tilde{k}$ (Komen et al., 1994).

#### *5.2.4.5.2 Triad Wave-wave Interactions*

The LTA of Eldeberky (1996), which is a slightly adapted version of the Discrete Triad Approximation (DTA) of Eldeberky and Battjes (1995), is used in SWAN in each spectral direction:

$$S_{nl3}(\sigma,\theta) = S_{nl3}^{-}(\sigma,\theta) + S_{nl3}^{+}(\sigma,\theta) \quad , \tag{37}$$

with

$$S_{nl3}^{+}(\sigma,\theta) = \max\left\{0, \alpha_{EB}\, 2\pi c c_g J^2 \left|\sin(\beta)\right| \left\{E^2(\sigma/2,\theta) - 2E(\sigma/2,\theta)\,E(\sigma,\theta)\right\}\right\} \quad , \tag{38}$$

and

$$S_{nl3}^{-}(\sigma,\theta) = -2S_{nl3}^{+}(2\sigma,\theta) \quad , \tag{39}$$

in which $\alpha_{EB}$ is a tunable proportionality coefficient. The biphase $\beta$ is approximated with

$$\beta = -\frac{\pi}{2} + \frac{\pi}{2}\tanh\left(\frac{0.2}{Ur}\right) \quad , \tag{40}$$

with *ursell* number $Ur$

$$Ur = \frac{g}{8\sqrt{2}\pi^2}\frac{H_s\overline{T}^2}{d^2} \quad , \tag{41}$$

where $\overline{T} = 2\pi/\overline{\sigma}$. The triad wave-wave interactions are calculated only for $10 > Ur > 0.1$. The interaction coefficient $J$ is taken from Madsen and Sørensen (1993):

$$J = \frac{k_{\sigma/2}^2(g\,d + 2c_{\sigma/2}^2)}{k_\sigma d(g\,d + \frac{2}{15}g\,d^3k_\sigma^2 - \frac{2}{5}\sigma^2 d^2} \quad . \tag{42}$$

#### 5.2.4.5.3 *Wave-induced Set-up*

In a (geographic) 1-D case the computation of the wave induced set-up is based on the vertically integrated momentum balance equation which is a balance between the wave force (gradient of the wave radiation stress normal to the coast) and the hydrodynamic pressure gradient (note that the component parallel to the coast causes wave-induced currents but no setup).

$$F_x + g\,d\frac{\partial\overline{\eta}}{\partial x} = 0 \tag{43}$$

where $d$ is the total water depth (including the wave-induced set-up) and $\eta$ is the mean surface elevation (including the wave-induced set-up).

Observation and computations based on the vertically integrated momentum balance equation of Dingemans et al. (1987) show that the wave-induced currents are mainly driven by the divergence-free part of the wave forces, whereas the set-up is mainly due to the rotation-free part of these forces. To compute the set-up, it would then be sufficient to consider the divergence of the momentum balance equation. If the divergence of the acceleration in the resulting equation is ignored, the result is:

$$\frac{\partial F_x}{\partial x}+\frac{\partial F_y}{\partial y}+\frac{\partial}{\partial x}\left(g\,d\frac{\partial \zeta}{\partial x}\right)+\frac{\partial}{\partial y}\left(g\,d\frac{\partial \zeta}{\partial y}\right)=0\ . \tag{44}$$

### *5.2.5 Numerical Implementation*

The integration of the action balance equation has been implemented in SWAN with finite difference schemes in all five dimensions, including time, geographic space and spectral space, etc. These are first described for the propagation of the waves without the source terms for generation, dissipation and wave-wave interactions. Then the implementation of these source terms is described.

Time is discretized with a simple constant timestep, $\Delta t$, for the simultaneous integration of the propagation and the source terms. This is different from how time was discretized in the WAM model or the WAVEWATCH III model where the timestep for propagation is different from the timestep for the source terms. Geographic space is discretized with a rectangular grid with constant resolutions $\Delta x$ and $\Delta y$ in $x$- and $y$-direction respectively (in fact, this rectangular grid is a special case of the curvilinear grid that has been programmed in SWAN). The spectrum in the model is discretized with a constant directional resolution $\Delta\theta$ and a constant relative frequency resolution $\Delta\sigma/\sigma$ (logarithmic frequency distribution). For reasons of economy, an option is available to compute only wave components traveling in a pre-defined directional sector ($\theta_{min} < \theta < \theta_{max}$; e.g., those components that travel shoreward within a limited directional sector). The discrete frequencies are defined between a fixed low-frequency cut-off and a fixed high-frequency cut-off (the prognostic part of the spectrum). For these frequencies the spectral density is unconstrained. Below the low-frequency cut-off (typically $f_{min}$ = 0.04 Hz for field conditions) the spectral densities are assumed to be zero. Above the high-frequency cut-off (typically 1 Hz for field conditions) a diagnostic tail $f^{-m}$ is added (this tail is used to compute nonlinear wave-wave interactions at the high frequencies and to compute integral wave parameters). The reason for using a fixed high-frequency cut-off rather than a dynamic cut-off frequency that depends on the wind speed or on the mean frequency, as in WAM and WAVEWATCH III, is that in coastal regions mixed sea states with rather different characteristic frequencies may occur. For instance, a local wind may generate a very young sea behind an island, totally unrelated to (but superimposed on) a simultaneously occurring swell. In such cases a dynamic cut-off frequency may be too low to properly account for the locally generated sea state. Based on physical arguments the value of $m$ (the power in the above expression of the spectral tail) should be between four and five (e.g., Phillips, 1985). In SWAN, $m = 4$ if the wind input formulation of Komen et al. (1984) is used (cf. WAM Cycle 3), and $m = 5$ if the wind input formulation of Janssen (1991a) is used (WAM Cycle 4).

#### 5.2.5.1 Propagation

The numerical schemes in SWAN have been chosen on the basis of robustness, accuracy and economy. Since the nature of the basic equation is such that the state in a grid point is determined by the state in the upwave grid points, the most robust scheme would be an implicit upwind scheme (in both geographic and spectral space). The adjective "implicit" is used here to indicate that all derivatives of action density (in t, x or y) are formulated at one computational level ($i_t$, $i_x$ or $i_y$) except the derivative in the integration dimension for which also the previous or upwave level is used (time in non-stationary mode and x or y in stationary mode). For such a scheme the values of the time and space steps $\Delta t$, $\Delta x$, and $\Delta y$ would be mutually independent. An implicit scheme would also be economical in the sense that such a scheme is unconditionally stable. It permits relatively large timesteps in the computations (much larger than for explicit schemes in shallow water). Several years of experience in using the second-generation HISWA shallow water wave model (Holthuijsen et al., 1989) has shown that for coastal regions a first-order upwind difference scheme in geographic space is usually accurate enough. This experience, together with test computations with SWAN has also shown that in spectral space a higher accuracy than that of a first-order upwind scheme is required. This can be achieved by supplementing such a scheme with a second-order central approximation (more economic than a second-order upwind scheme). For SWAN therefore, implicit upwind schemes in both geographic and spectral space have been chosen, supplemented with a central approximation in spectral space.

The fact that in geographic space, the state in a grid point is determined by the state in the upwave grid points (as defined by the direction of propagation), permits a decomposition of the spectral space into four quadrants (eight octants would be an alternative). In each of the quadrants the computations can be carried out independently from the other quadrants except for the interactions between them due to refraction and nonlinear wave-wave interactions (formulated in corresponding boundary conditions between the quadrants). The wave components in SWAN are correspondingly propagated in geographic space with an upwind scheme (upwind is the common term in numerical analysis, but up-wave would be more appropriate in the case of SWAN). SWAN contains three such schemes:

a. First-order (stationary and non-stationary cases) backward space-backward time (BSBT) scheme,
b. Second-order (non-stationary cases) with third-order diffusion: the S&L scheme (Stelling and Leedertse, 1992),
c. Second-order (stationary cases) with second-order diffusion (SORDUP) scheme.

The BSBT scheme (not default in SWAN) will be discussed first and then the extension to the higher order schemes that are default in SWAN. The first-order upwind scheme (BSBT) is a sequence of four forward-marching sweeps (one per quadrant). To properly account for the boundary conditions between the four quadrants, the computations are carried out iteratively at each timestep. The integration in time is a simple backward finite difference, so that the discretization of the action balance equation is (for positive propagation speeds; including the computation of the source terms but ignoring their discretization):

$$\left[\frac{N^{i_t,n}-N^{i_t-1}}{\Delta t}\right]_{i_x,i_y,i_\sigma,i_\theta} + \left[\frac{[c_xN]_{i_x} - [c_xN]_{i_x-1}}{\Delta x}\right]^{i_t,n}_{i_y,i_\sigma,i_\theta} + \left[\frac{[c_yN]_{i_y} - [c_yN]_{i_y-1}}{\Delta y}\right]^{i_t,n}_{i_x,i_\sigma,i_\theta}$$
$$+\left[\frac{(1-\nu)[c_\sigma N]_{i_\sigma+1} + 2\nu[c_\sigma N]_{i_\sigma} - (1+\nu)[c_\sigma N]_{i_\sigma-1}}{2\Delta\sigma}\right]^{i_t,n}_{i_x,i_y,i_\theta} +$$
$$\left[\frac{(1-\eta)[c_\theta N]_{i_\theta+1} + 2\eta[c_\theta N]_{i_\theta} - (1+\eta)[c_\theta N]_{i_\theta-1}}{2\Delta\theta}\right]^{i_t,n}_{i_x,i_y,i_\sigma} = \left[\frac{S}{\sigma}\right]^{i_t,n^*}_{i_x,i_y,i_\sigma,i_\theta} \tag{45}$$

where $i_t$ is the time-level index and $i_x$, $i_y$, $i_\sigma$ and $i_\theta$ are grid counters and $\Delta t$, $\Delta x$, $\Delta y$, $\Delta\sigma$, and $\Delta\theta$ are the increments in time, geographic space and spectral space respectively. The iterative nature of the computation is indicated with the iteration index $n$ (the iteration index for the source terms $n^*$ is equal to $n$ or $n$-1, depending on the source term, see below). Because of these iterations, the scheme is also approximately implicit for the source terms. For negative propagation speeds, appropriate + and - signs are required in Eq. 45.

The coefficients $\nu$ and $\eta$ determine the degree to which the scheme in spectral space is upwind or central. They control the numerical diffusion in frequency and directional space, respectively. A value of $\nu = 0$ or $\eta = 0$ corresponds to central schemes which have the largest accuracy (numerical diffusion $\approx 0$). Value of $\nu = 1$ or $\eta = 1$ correspond to upwind schemes which are somewhat more diffusive and therefore less accurate but more robust. If large gradients of the action density in frequency space or directional space are present, numerical oscillations can arise (especially with the central difference schemes) resulting in negative values of the action density. In each sweep such negative values are removed from the two-dimensional spectrum by setting these values equal to zero and rescaling the remaining positive values such that the frequency-integrated action density per spectral direction is conserved. The depth derivatives and current derivatives in the expressions of $c_\sigma$ and $c_\theta$ are calculated with a first-order upwind scheme. For very strong refraction the value of $c_\theta$ is reduced in each grid point and for each wave component individually with the square of the fraction of the grid spacing over which $kd < 3.0$.

For stationary conditions SWAN can be run in stationary mode. Time is then removed as a variable but the integration (in geographic space) is still carried out iteratively. The propagation scheme is still implicit as the derivatives of action density (in x or y) at the computational level ($i_x$ or $i_y$, respectively) are formulated at that level except in the integration dimension (x or y; depending on the direction of propagation) where the upwave level is used. The values of $\Delta x$ and $\Delta y$ are therefore still mutually independent.

For the stationary second-order upwind scheme (Rogers et al., 2000; SORDUP) which is the default scheme for stationary computations, the two terms in Eq. 45 representing x- and y-derivatives are replaced by:

$$\left[\frac{1.5[c_x N]_{i_x} - 2[c_x N]_{i_x-1} + 0.5[c_x N]_{i_x-2}}{\Delta x}\right]^{i_t,n}_{i_y,i_\sigma,i_\theta} +$$

$$\left[\frac{1.5[c_y N]_{i_y} - 2[c_y N]_{i_y-1} + 0.5[c_y N]_{i_y-2}}{\Delta y}\right]^{i_t,n}_{i_x,i_\sigma,i_\theta} \qquad (45a)$$

For the non-stationary second-order upwind scheme (Rogers et al., 2000; S&L), which is the default scheme for non-stationary computations, the two terms in Eq. 45 representing x- and y-derivatives are replaced by:

$$\left[\frac{\frac{5}{6}[c_x N]_{i_x} - \frac{5}{4}[c_x N]_{i_x-1} + \frac{1}{2}[c_x N]_{i_x-2} - \frac{1}{12}[c_x N]_{i_x-3}}{\Delta x}\right]^{i_t,n}_{i_y,i_\sigma,i_\theta} +$$

$$\left[\frac{\frac{5}{6}[c_y N]_{i_y} - \frac{5}{4}[c_y N]_{i_y-1} + \frac{1}{2}[c_y N]_{i_y-2} - \frac{1}{12}[c_y N]_{i_y-3}}{\Delta y}\right]^{i_t,n}_{i_x,i_\sigma,i_\theta} + \qquad (45b)$$

$$\left[\frac{\frac{1}{4}[c_x N]_{i_x+1} - \frac{1}{4}[c_x N]_{i_x-1}}{\Delta x}\right]^{i_t-1}_{i_y,i_\sigma,i_\theta} + \left[\frac{\frac{1}{4}[c_y N]_{i_y+1} - \frac{1}{4}[c_y N]_{i_y-1}}{\Delta y}\right]^{i_t-1}_{i_x,i_\sigma,i_\theta}$$

To explain the above numerical solution technique in terms of matrix solutions, first ignore the decomposition in quadrants. The propagation of the waves in both geographic and spectral space would then be described with one large basic matrix that can be solved in several ways. Removing refraction, frequency shifting and nonlinear source terms from this basic matrix permits a matrix solution with a Gauss-Seidel technique (e.g., Golub and van Loan, 1986) in which the matrix is decomposed in four sections (the above four directional quadrants) which are each solved in one step (super-convergence). Restoring refraction and frequency shifting to the matrix requires the solution of a submatrix for each geographic grid point. If no currents are present and the depth is stationary, this is readily done with a Thomas algorithm (e.g., Abbott and Basco, 1989; $c_\sigma = 0$ and the sub-matrix is a simple tri-diagonal matrix). If currents are present or the depth is not stationary, the sub-matrix is a band matrix. It is solved with an iterative ILU-CGSTAB method (Vuik, 1993; Van der Vorst, 1992). Restoring refraction and frequency-shifting also introduces coefficients in each matrix section (directional quadrant) that cause dependency between the matrix sections. The same happens when nonlinear source terms are added to the matrix. The basic matrix as a whole therefore needs to be solved iteratively until some break-off criteria are met. To reduce the number of iterations in stationary mode with wind generation, SWAN starts with a reasonable first-guess of the wave field (a "quickstart" based on the second-generation source terms of Holthuijsen and De Boer, (1988) adapted for shallow water). It reduces the number of iterations typically by a factor two. In non-stationary mode, a very reasonable first-guess per timestep is available from the previous timestep and the number of iterations is expected to be small. If no iterations are used in non-stationary mode (as in most phase averaged wave models), the computations of propagation are still implicit and therefore still unconditionally stable.

In the neighborhood of grid points which represent open boundaries, land boundaries and obstacles (i.e., the last two grids adjoining such grid points for the SORDUP scheme and the last three grids adjoining such grid points for the S&L scheme), SWAN will revert to the first-order BSBT scheme. This scheme has a larger numerical diffusion but that is usually acceptable over the small distances involved.

The numerical diffusion of the S&L scheme is so small that the so-called garden-sprinkler effect (GSE) may show up if propagation over very large distances is considered. This effect is due to the spectral resolution (Booij and Holthuijsen, 1987). It can be counteracted by a diffusion term that has been explicitly added to the numerical scheme (not default in SWAN). Its value depends on the spectral resolution and the propagation time of the waves (see the input variable [wave age] in the SCHEME command).

The diffusion applied in the propagation direction is:

$$D_{ss} = \Delta c^2 T / 12 \quad , \tag{46}$$

where $T$ is the wave age.

The diffusion normal to the propagation direction is:

$$D_{ss} = c^2 \Delta\theta^2 T / 12 \quad . \tag{47}$$

From these diffusion coefficients (in terms of x and y) are calculated:

$$D_{xx} = D_{ss}\cos^2\theta + D_{nn}\sin^2\theta;$$
$$D_{yy} = D_{ss}\sin^2\theta + D_{nn}\cos^2\theta; \qquad (48)$$
$$D_{xy} = (D_{ss} - D_{nn})\cos\theta\sin\theta\,.$$

The diffusion terms are computed at the time level $i_t$ - 1. The diffusion terms are computed as follows:

$$\begin{gathered}
D_{xx}\left[\frac{[N]_{i_x+1} - 2[N]_{i_x} + [N]_{i_x-1}}{\Delta x^2}\right]^{i_t-1}_{i_y,i_\sigma,i_\theta} \\
D_{yy}\left[\frac{[N]_{i_y+1} - 2[N]_{i_y} + [N]_{i_y-1}}{\Delta y^2}\right]^{i_t-1}_{i_x,i_\sigma,i_\theta} \\
D_{xy}\left[\frac{[N]_{i_x,i_y} - [N]_{i_x-1,i_y} - [N]_{i_x,i_y-1} + [N]_{i_x-1,i_y-1}}{\Delta x\Delta y}\right]^{i_t-1}_{i_\sigma,i_\theta}
\end{gathered} \qquad . \qquad (49)$$

This explicit finite differentiation is fast (having little impact on computation time) but only conditionally stable. Through mathematical analysis (not shown) it can be shown that a likely stability condition for the one-dimensional S&L scheme with this GSE correction is $D\Delta t/(\Delta x^2) \leq 0.5$ which corresponds to the two-dimensional stability criterion of Tolman (1995); (based on Fletcher, 1988):

$$Q = \frac{\max(D_{xx}, D_{yy}, D_{xy})\Delta t}{\min(\Delta x\Delta y)^2} \leq 0.5 \qquad . \qquad (50)$$

Thus it is credible that Eq. 50 holds true for the two-dimensional S&L scheme with this GSE correction. In experiments, it was found that for all experiments which satisfy the slightly more restrictive $Q \leq 0.48$ instability was observed. In short, by adding the GSE correction, the unconditionally stable advection scheme of SWAN becomes a (likely) conditionally stable advection diffusion scheme. It is readily shown that for typical ocean applications $D_{nn}$ dominates the diffusion and can be written as:

$$Q = \overline{C}T/\Delta x.\overline{C}\Delta t/\Delta x.\Delta\theta^2/12 \qquad . \qquad (51)$$

The variable wave age $\overline{T}$ could be computed during the computations of SWAN (Booij and Holthuijsen, 1987) but it requires the same order of magnitude of computer memory as integrating the action balance equation. Instead a constant wave age $\overline{T}$ can be used as an approximation, so that Eq. 51 becomes

$$Q = \overline{L}/\Delta x.\mu.\Delta\theta/12 \qquad , \qquad (52)$$

where the characteristic travel distance of the waves is $\overline{L} = \overline{C}\overline{T}$ (e.g., the dimension of the ocean basin). For oceanic applications the Courant number is typically $\mu \approx ½$ so that $Q \leq 0.25$ for typical values of $\Delta\theta$ and $\overline{L}/\Delta x$ (the number of grid point in one direction of the grid). This implies that the S&L scheme with this GSE correction is stable for typical ocean cases. For shelf sea (regional) applications the value of $\mu = O(1)$ but the garden-sprinkler effect tends to be small on these scales and the diffusion can and should not be used to avoid the stability problem. For small-scale (local) applications typically $\mu = O(10\text{-}100)$. But such cases are usually treated as stationary and the SORDUP scheme should be used (no GSE correction is included in this scheme).

The boundary conditions in SWAN, both in geographic space and spectral space are fully absorbing for wave energy that is leaving the computational domain or crossing a coastline. The incoming wave energy along open geographic boundaries needs to be prescribed by the user. For coastal regions such incoming energy is usually provided only along the deep-water boundary and not along the lateral geographic boundaries (i.e., the spectral densities are assumed to be zero). This implies that such erroneous lateral boundary conditions are propagated into the computational area. The affected areas are typically triangular regions with the apex at the corners, between the deep-water and lateral boundaries, spreading towards shore at an angle of 30° to 45° (for wind sea conditions) on either side of the deep-water mean wave direction (less for swell conditions; this angle is essentially equal to the one-sided width of the directional distribution of the incoming wave spectrum). For this reason the lateral boundaries should be sufficiently far away from the area of interest to avoid the propagation of this error into the area.

##### *5.2.5.1.1 Generation, Wave-wave Interactions and Dissipation*

The numerical estimations of the source terms in SWAN are essentially implicit. This is achieved with explicit or implicit approximations of the source terms which in the limit of a large number of iterations, always result in an implicit estimation. In actual computations final convergence is obviously never achieved and the estimations of the source terms are therefore strictly speaking only approximately implicit. In the following, "explicit" and "implicit" refer to the approximations of the source terms within each iteration.

The linear growth term $A$ is independent of integral wave parameters and of the energy density and can therefore be readily computed. All other source terms depend on energy density and they can be described as a (quasi-) linear term: $S = \phi E$, in which $\phi$ is a coefficient that depends on (integral) wave parameters (e.g., $E_{tot}$, $\tilde{\sigma}$, $\tilde{k}$, $\sigma$, $k$, etc.) and action densities of other spectral components. Since these are only known at the previous iteration level $n$-1, the coefficient is determined at that iteration level: $\phi = \phi^{n-1}$.

For positive source terms (wind input and the triad wave-wave interactions if positive) the integration is generally more stable if an explicit formulation is used (i.e., the source term depends on $E^{n-1}$ and not on $E^n$) rather than an implicit formulation (i.e., the source term also depends on $E^n$). The explicit formulation for these source terms in SWAN is therefore:

$$S^n \approx \phi^{n-1} E^{n-1} \quad . \tag{53}$$

For reasons of economy this explicit approximation is also used for the formulation of the quadruplet wave-wave interactions (for both the positive and negative contributions). This is considered reasonable since Tolman (1992a) has shown that using an explicit formulation in combination with a limiter (see below) gives similar results as the use of a more expensive implicit scheme (this implicit formulation is optionally available in SWAN; in the WAM model it is indicated as the semi-implicit scheme, (the WAMDI group, (1988); Komen et al, (1994))).

For negative source terms the integration is generally more stable if an implicit scheme is used. The strongly nonlinear, negative source term of depth-induced wave breaking at iteration level $n$ is accordingly estimated with a linear approximation:

$$S^n \cong \phi^{n-1} E^{n-1} + \left(\frac{\partial S}{\partial E}\right)^{n-1} (E^n - E^{n-1}) \quad . \tag{54}$$

However, to achieve even more stable computations for this source term, the term $\phi^{n-1}E^{n-1}$ in this formulation has been replaced by $\phi^{n-1}E^n$ (making the formulation somewhat more implicit and thus more robust; note that in the limit the solution is the same). Since this process of depth-induced wave breaking has been formulated such that $S = aS_{tot}$ and $E = aE_{tot}$, the derivative is $\partial S / \partial E$ analytically determined as $\partial S_{tot} / \partial E_{tot}$ (where is $a$ identical in both expressions and the total energy $E_{tot}$ and the total source $S_{tot}$ are the integrals over all frequencies and directions of $E(\sigma, \theta)$ and $S_{ds,br}(\sigma, \theta)$, respectively). For the other negative (mildly nonlinear) source terms, i.e., whitecapping, bottom friction and negative triad wave-wave interactions, a similar accuracy of estimating $S^n$ can be achieved with the following simpler, and therefore more economical approximation in which $(\partial S / \partial E)^{n-1}$ of Eq. 14 has been replaced by $(S/E)^{n-1}$

$$S^n \cong \phi^{n-1} E^{n-1} + \left(\frac{S}{E}\right)^{n-1} (E^n - E^{n-1}) \quad . \tag{55}$$

With $S = \phi E$, this reduces to:

$$S^n \cong \phi^{n-1} E^n \quad . \tag{56}$$

These approximations for the source terms are added to the elements of the matrix for the propagation. To suppress the development of numerical instabilities, the maximum total change of action density per iteration at each discrete wave component is limited to a fraction of 10% of the Phillips (1957) equilibrium level (reformulated in terms of action density and wave number to be applicable in shallow water; as in the WAM model and in the WAVEWATCH III model (Tolman, 1992a)):

$$\left|\Delta N(\sigma,\theta)\right|_{\max} = \frac{0.1}{2\pi\sigma}\frac{\alpha_{PM}\pi}{k^3 c} \quad , \tag{57}$$

where $\alpha_{PM} = 0.0081$ is the Phillips' "constant" of the Pierson-Moskowitz (1964) spectrum. To retain the very rapid but realistic decrease of wave energy near the shore due to depth-induced wave breaking, this limiter is not applied if the waves actually break (in SWAN: $H_{rms}/H_{\max} < 0.2$ with $H_{rms} = \sqrt{8E_{tot}}$ which implies a fraction of breakers $Q_b >$ 0.00001).

The fraction of depth-induced breakers ($Q_b$) is determined in SWAN with

$$Q_b = 0 \qquad \text{for} \quad \beta \le 0.2$$

$$Q_b = Q_0 - \beta^2 \frac{Q_0 - \exp((Q_0 - 1)/\beta^2)}{\beta^2 - \exp((Q_0 - 1)/\beta^2)} \qquad \text{for} \quad 0.2 < \beta < 1 \tag{58}$$

$$Q_b = 1 \qquad \text{for} \quad \beta \ge 1$$

where $\beta = H_{rms}/H_{\max}$ . For $\beta \le 0.5$, $Q_0 = 1$, and for $0.5 < \beta \le 1$, $Q_0 = (2\beta - 1)^2$.

#### *5.2.5.1.2 Wave-induced Set-up*

In 1-D cases the wave-induced set-up is calculated in SWAN with a simple trapezoidal rule.

In 2-D cases the Poisson equation of the divergence-free force field is solved in SWAN with the same solver that is used for wave propagation with ambient currents. The boundary conditions for this elliptical partial differential equation are:

Non-nested computations:

- At open boundaries, the equilibrium between wave force and hydrodynamic pressure gradient is normal to the model boundary,
- At last grid points before shoreline, the equilibrium between wave force and hydrodynamic pressure gradient is normal to the model boundary,
- At deepest boundary point, the set-up is zero.

Nested computations:

- At open boundaries, the set-up is taken from the larger computation,
- At last grid points before shoreline, the equilibrium between wave force and hydrodynamic pressure gradient is normal to the model boundary.

The shoreline in SWAN moves as dictated by the wave-induced set-up. The set-up computations are available on both the rectilinear and curvilinear grids.

#### *5.2.5.1.3 Curvilinear Grid*

The propagation scheme in SWAN for geographic space is formulated on a curvilinear geographic grid (irregular, quadrangular, and not necessarily orthogonal) rather than the rectilinear grid of SWAN Cycle I. This modification is based on approximating the geographic distribution of the energy (action) density between each three neighboring grid points with a flat triangle. The gradient in each grid point at location ($x_i$, $y_j$) is then readily approximated from the up-wind grid points. For the *x*-direction this approximation is for grid point *i, j* (the grid points are ordered in *x, y*-space with labels *i* and *j* respectively):

$$\frac{\partial C_x N}{\partial x} \cong \left[\frac{[c_x N]_{i,j} - [c_x N]_{i-1,j}}{\Delta\tilde{x}_1}\right] + \left[\frac{[c_x N]_{i,j} - [c_x N]_{i,j-1}}{\Delta\tilde{x}_2}\right] \quad , \tag{59}$$

where $\Delta\tilde{x}_1 = \Delta x_1 - (\Delta y_1 / \Delta y_2)\Delta x_2$, $\Delta\tilde{x}_2 = \Delta x_2 - (\Delta y_2 / \Delta y_1)\Delta x_1$. The increments are $\Delta x_1 = x_{i,j} - x_{i-1,j}$, $\Delta x_2 = x_{i,j} - x_{i,j-1}$, $\Delta y_1 = y_{i,j} - y_{i-1,j}$ and $\Delta y_2 = y_{i,j} - y_{i,j-1}$. The gradient in y-direction is similarly estimated.

### *5.2.6 SWAN Physics*

SWAN accounts for the following Physics:

- wave propagation in time and space, shoaling, refraction due to current and depth, frequency shifting due to currents and non-stationary depth;
- wave generation by wind;
- three- and four-wave interactions;
- whitecapping, bottom friction, and depth-induced breaking;
- wave induced setup;
- propagation from laboratory up to global scales;
- transmission through and reflection from obstacles.

# 5.3 SWAN ROUTINES

## *5.3.1 Command Reading Routines (ocpcre FOR File)*

### 5.3.1.1 Logical Function EQCSTR

Function EQCSTR is assigned the value True if the two strings are the same (case-insensitive).

| | | |
|---|---|---|
| **Calling Sequence:** | eqcstr (str1, str2) | |
| **Data Declaration:** | Character | str1, str2 |
| **Arguments:** | str1, str2 | Two character strings to be compared. |

### 5.3.1.2 Subroutine GETKAR

Subroutine GETKAR reads the next character (KAR) from the string KAART. The position of this character in KAART is indicated by KARNR. If needed, a new input line is read. At the end of the input file, ELTYPE is made EOF.

### 5.3.1.3 Subroutine IGNORE

Subroutine IGNORE calls subroutine INKEYW to read a keyword. If this keyword is equal to *string*, ELTYPE is made USED. It is used if a keyword can occur in the input, which does not lead to any action.

| | | |
|---|---|---|
| **Calling Sequence:** | ignore (string) | |
| **Data Declaration:** | Character | string |
| **Arguments:** | string | Keyword (if appearing in the input file) that can be ignored. |

### 5.3.1.4 Subroutine INCSTR

Subroutine INCSTR reads a string in free format.

| | | |
|---|---|---|
| **Calling Sequence:** | incstr (naam, c, kont, csta) | |
| **Data Declaration:** | Character | naam, c, kont, csta |

| | | |
|---|---|---|
| **Arguments:** | naam | Name of the variable according to the User's Manual. |
| | kont | Variable options: |
| | | = req Error message if no value is found in the input file; |
| | | = unc If no value, then variable will not be changed; |
| | | = sta If no value, then variable will get default value; |
| | | = rqi Variable may not have the value of *csta*; |
| | | = rep Repeat; |
| | | = nskp No skip. If data item is of different type, value is left unchanged. |
| | c | String to be read from the input file. |
| | csta | Default value of the string. |

### 5.3.1.5 Subroutine INCTIM

Subroutine INCTIM reads and interprets a time string.

**Calling Sequence:** inctim (ioptim, naam, rv, kont, rsta)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | ioptim |
| | Real | rv, rsta |
| | Character | naam, kont |

| | | |
|---|---|---|
| **Arguments:** | ioptim | Time reading option (see subroutine DTSTTI). |
| | rv | Variable that is to be assigned a value. |
| | rsta | Default value. |
| | naam | Name of the variable according to the User's Manual. |
| | kint | Variable options: |
| | | = req Error message if no value is found in the input file; |
| | | = unc If no value, then variable will not be changed; |
| | | = sta If no value, then variable will get default value; |
| | | = rqi Variable may not have the value of *rsta*; |
| | | = rep Repeat; |
| | | = nskp No skip. If data item is of different type, value is left unchanged. |

### 5.3.1.6 Subroutine INDBLE

Subroutine INDBLE reads a double precision number in free format.

| | | |
|---|---|---|
| **Calling Sequence:** | indble (naam, r, kont, rsta) | |
| **Data Declaration:** | Real | r, rsta |
| | Character | naam, kont |
| **Arguments:** | r | The value of the variable that is to be read. |
| | rsta | Reference value needed for *kont* = sta or rqi |
| | kont | Variable options:<br>= req Variable is required;<br>= unc If no variable, then variable will not be changed;<br>= sta If no variable, then variable will get value of *rsta*;<br>= rqi Variable may not have the value of *rsta*;<br>= rep Repeat;<br>= nskp No skip. If the data item is of a different type then the value is left unchanged. |
| | naam | Name of the variable according to the User's Manual. |

### 5.3.1.7 Subroutine ININTG

Subroutine ININTG reads an integer number in free format.

| | | |
|---|---|---|
| **Calling Sequence:** | inintg (naam, iv, kont, ista) | |
| **Data Declaration:** | Integer | iv, ista |
| | Character | kont, naam |
| **Arguments:** | iv | Integer variable that is to be assigned a value. |
| | ista | Default value. |
| | naam | Name of the variable according to the User's Manual. |
| | kont | Variable options:<br>= req Error message if no value is found in the input file;<br>= unc If no value, then variable will not be changed;<br>= sta If no value, then variable will get default value; |

= rqi Variable may not have the value of *rsta*;
= rep Repeat;
= nskp No skip. If the data item is of a different type, then the value is left unchanged.

### 5.3.1.8 Subroutine ININTV

Subroutine ININTV reads a time interval in the form: number day/hr/min/sec.

**Calling Sequence:** inintv (naam, rvar, kont, rsta)

**Data Declaration:**

| | |
|---|---|
| Character | kont, naam |
| Real | rvar, rsta |

**Arguments:**

naam — Name of the variable according to the User's Manual.

kont — Variable options:
= req Error message if no value is found in the input file;
= unc If no value, then variable will not be changed;
= sta If no value, then variable will get default value;
= rqi Variable may not have the value of *rsta*;
= rep Repeat;
= nskp No skip. If data item is of a different type, value is left unchanged.

rsta — Default value.

rvar — Variable that is to be assigned a value.

### 5.3.1.9 Subroutine INKEYW

Subroutine INKEYW reads a keyword.

**Calling Sequence:** inkeyw (kont, csta)

**Data Declaration:** Character kont, csta

**Arguments:**

kont — Action to be taken if no keyword is found in input:
= req Required. Error message;
= sta Standard. The value of *csta* is assigned to keyword.

csta — Default value of the string.

### 5.3.1.10 Subroutine INREAL

Subroutine INREAL reads a real number in free format.

**Calling Sequence:** inreal (naam, r, kont, rsta)

**Data Declaration:**

| | |
|---|---|
| Real | r, rsta |
| Character | naam, kont |

**Arguments:**

| | |
|---|---|
| r | The value of the variable that is to be read. |
| rsta | Reference value needed for *kont* = sta or rqi. |
| kont | Variable options:<br>= req Variable is required;<br>= unc If no variable, then variable will not be changed;<br>= sta If no variable, then variable will get value of *rsta*;<br>= rqi Variable may not have the value of *rsta*;<br>= rep Repeat;<br>= nskp No skip. If data item is of different type, then the value is left unchanged. |
| naam | Name of the variable according to the User's Manual. |

### 5.3.1.11 Subroutine KEYWIS

Function KEYWIS tests whether or not a keyword given by the user coincides with a keyword known in the program (i.e. *string*). If so, KEYWIS is made True, otherwise it is False. ELTYPE is made USED, so that the next element can be read.

**Calling Sequence:** keywis (string)

**Data Declaration:** Character string

**Arguments:** string A keyword, which is compared with another keyword found in the input file.

### 5.3.1.12 Subroutine LEESEL

Subroutine LEESEL reads a new data item from the string KAART. The type of the item is determined, and the contents appear in ELTEXT, ELINT, or ELREAL, as the case may be.

The following types are distinguished:

| | |
|---|---|
| KEY | Keyword. |
| INT | Integer or real number. |
| REAL | Real number. |
| CHAR | Character string enclosed in quotes. |
| EMPT | Empty data field. |
| OTHR | Non-empty data item not recognized as real, integer or character. Possibly a time string. |
| EOF | End of input file. |
| EOR | End of repeat or end of record. |
| ERR | Error. |
| USED | Used, item last read is processed already. |

### 5.3.1.13 Subroutine NWLINE

Subroutine NWLINE jumps to the reading of the next input line if the end of the previous one is reached.

### 5.3.1.14 Subroutine PUTKAR

Subroutine PUTKAR inserts a character (*karr*), usually read by subroutine GETKAR, into the string *ltext*, which is equal to ELTEXT, in the place *jkar*. After this, *jkar* is increased by one.

**Calling Sequence:** putkar (ltext, karr, jkar)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | jkar |
| | Character | karr, ltext |

| | | |
|---|---|---|
| **Arguments:** | jkar | Counts the number of characters in a data field. |
| | ltext | Character string. After a number of calls it will contain the character representation of a data field. |
| | karr | Character to be inserted into *ltext*. |

#### 5.3.1.15 Subroutine RDINT

Subroutine RDINT initializes the command reading system.

#### 5.3.1.16 Subroutine UPCASE

Subroutine UPCASE changes all characters of the string *charst* from lower to uppercase.

| | | |
|---|---|---|
| **Calling Sequence:** | upcase (charst) | |
| **Data Declaration:** | Character | charst |
| **Arguments:** | charst | A character string. |

#### 5.3.1.17 Subroutine WRNKEY

Subroutine WRNKEY produces an error message. It is called if an illegal keyword is found in the user's input. It makes ELTYPE = USED.

### *5.3.2 Dynamic Data Pool Routines (ocpdpn FOR Files)*

#### 5.3.2.1 Subroutine COPYCH

Subroutine COPYCH copies a string into an integer array or vice-versa. The variable *move* (TO_ or FROM_) indicates the copying direction.

| | | |
|---|---|---|
| **Calling Sequence:** | copych (string, move, iarray, lenarr, ierr) | |
| **Data Declaration:** | Integer | iarray, lenarr, ierr |
| | Character | string, move |
| **Arguments:** | iarray | An integer *array*. |
| | lenarr | Length of *iarray*. |
| | ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |
| | string | A character string. |
| | move | If *move* = to_, *string* is copied to *iarray*;<br>If *move* = from_, *string* is copied from *iarray*. |

### 5.3.2.2 Subroutine DPADDP

Subroutine DPADDP adds a new pointer. If the name of the pointer is not yet present, all of the data in *array* after the names and pointers of the existing point-sets are moved *lenpnt* places. The free places are then filled with the new name, which is the pointer to the start of the record and the record length.

| | | |
|---|---|---|
| **Calling Sequence:** | dpaddp (array, pname, pindex, ptype, padres, ierr) | |
| **Data Declaration:** | Integer | array, pindex, ptype, padres, ierr |
| | Character | pname |
| **Arguments:** | array | Array in which the pointer structure exists. |
| | ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |
| | padres | Location in *array* of first data. |
| | pindex | Index of the new pointer. |
| | ptype | Type of data referenced by the new pointer:<br>S = Single precision data;<br>P = Pointers of the record referenced by the pointer. |
| | pname | Name of the new pointer. |

### 5.3.2.3 Subroutine DPBLDP

Subroutine DPBLDP builds a *pool* structure into *array*.

| | | |
|---|---|---|
| **Calling Sequence:** | dpbldp (array, lenarr, lenpnm, lenadt, ierr) | |
| **Data Declaration:** | Integer | array, lenarr, lenpnm, lenadt, ierr |
| **Arguments:** | array | Array into which the pointer structure is to be built. |
| | lenarr | Length of *array*. If the input value is negative, it is assumed that the array already contains the proper length. |
| | lenpnm | Length provided for the names of the pointers. |
| | lenadt | Length provided for additional data in the pointer. |
| | ierr | Input:<br>If = 0 Standard message;<br>If = -1 No message;<br>If < -1 More complete message.<br>Output:<br>= 0 No errors, otherwise: > 0;<br>= 9 End-of-file. |

### 5.3.2.4 Subroutine DPCHEK

Subroutine DPCHEK checks the data integrity in the *pool* and displays the *pool* structure. *pool* cycles have to remain intact. Pointer index → record address → record length → end of the record. At the end of the record the pointer index must be found.

**Calling Sequence:** dpchek (array, ierr)

**Data Declaration:** Integer array, ierr

| | | |
|---|---|---|
| **Arguments:** | array | Array in which the pointer structure exists. |
| | ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |

### 5.3.2.6 Subroutine DPEXPR

Subroutine DPEXPR makes record number *pindex* the length *newsiz*. If the data type is real/integer then the return record address is *padres*. If the record data type is pointer, the *pool* structure is possibly destroyed if the record is reduced in length.

**Calling Sequence:** dpexpr (array, pindex, newsiz, padres, ierr)

**Data Declaration:** Integer array, pindex, newsiz, padres, ierr

| | | |
|---|---|---|
| **Arguments:** | array | Array in which the pointer structure exists. |
| | ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |
| | newsiz | New size of the record referenced by the pointer. |
| | padres | Location in *array* of the first data of the record referenced by the pointer. |
| | pindex | Index of a pointer. |

### 5.3.2.7 Integer Function DPGETI

Function DPGETI gives the integer value of element *pplace* of record number *pindex* in *array*.

**Calling Sequence:** dpgeti (array, pindex, pplace, ierr, move)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | array, pindex, pplace, ierr |
| | Character | move |
| **Arguments:** | array | Array in which the pointer structure exists. |
| | pindex | Index of the pointer. |
| | pplace | Number of elements in the record. |
| | ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |
| | move | If *move* = up, *pplace* is increased by one. |

### 5.3.2.9 Subroutine DPINQA

Subroutine DPINQA provides information about the base pointer of an array.

**Calling Sequence:** dpinqa (array, lenarr, lenocp, numpns, lenpnm, lenadt, ierr)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | array, lenarr, lenocp, numpns, lenpnm, lenadt, ierr |
| **Arguments:** | array | Array in which the pointer structure exists. |
| | lenarr | Length of *array*. |
| | lenocp | Number of occupied places in the array. |
| | numpns | Number of pointers in the array. |
| | lenpnm | Length provided for the names of the pointers. |
| | lenadt | Length provided for additional data in the pointer. |
| | ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |

### 5.3.2.10 Subroutine DPINQP

Subroutine DPINAP provides the index of a pointer given by name, as well as the address and length of the associated record. If the name of the pointer is not yet present, the index and address will both be made zero.

**Calling Sequence:** dpinqp (array, pname, pindex, ptype, padres, lenrec, ierr)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | array, pindex, padres, lenrec, ierr |
| | Character | pname, ptype |
| **Arguments:** | array | Array in which the pointer structure exists. |
| | pindex | Index of a pointer given by its name. |
| | padres | Location in *array* of the first data of the |

| | |
|---|---|
| | record referenced by the pointer. |
| lenrec | Length of the record referenced by the pointer. |
| ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |
| pname | Name of a pointer. |
| ptype | Type of data in record referenced by the pointer. |

### 5.3.2.11 Subroutine DPMAXR

Subroutine DPMINR makes record number *pindex* as long as possible. The length of the record is returned in *newsiz*. If the data type is real/integer the record address *padres* is returned.

**Calling Sequence:** dpmaxr (array, pindex, newsiz, padres, ierr)

**Data Declaration:** Integer array, pindex, newsiz, padres, ierr

**Arguments:**

| | |
|---|---|
| array | Array in which the pointer structure exists. |
| pindex | Index of a pointer. |
| newsiz | New size of the record referenced by the pointer. |
| padres | Location in *array* of the first data of the record referenced by the pointer. |
| ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |

### 5.3.2.12 Subroutine DPMINR

Subroutine DPSHFT makes record number *pindex* the length *newsiz*. If data type is real/integer then record address *padres* is returned. If the record data type is *pointer*, the *pool* structure is possibly destroyed if the record is reduced in length.

**Calling Sequence:** dpminr (array, pindex, newsiz, padres, ierr)

**Data Declaration:** Integer array, pindex, newsiz, padres, ierr

**Arguments:**

| | |
|---|---|
| array | Array in which the pointer structure exists. |
| ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |
| newsiz | New size of the record referenced by the pointer. |

| | |
|---|---|
| padres | Location in *array* of the first data. |
| pindex | Index of a pointer of the record referenced by the pointer. |

### 5.3.2.13 Subroutine DPPUTR

Subroutine DPPUTR puts a real value into an integer *array*. *Array* is declared here as real, but it is integer in the calling program.

**Calling Sequence:** dpputr (array, pplace, rv)

**Data Declaration:**

| | |
|---|---|
| Integer | pplace |
| Real | array, rv |

**Arguments:**

| | |
|---|---|
| pplace | Number of elements in *array*. |
| array | Array in which the pointer structure exists. |
| rv | Real variable to be put into *array*. |

### 5.3.2.14 Subroutine DPSHFT

Subroutine DPEXPR adds *mshif* to empty places (*mshif* > 0) or deletes -*mshif* places (*mshif* < 0) after ILOX in array IOUTD.

**Calling Sequence:** dpshft (array, linsrt, mshif, ierr)

**Data Declaration:** Integer array, linsrt, mshif, ierr

**Arguments:**

| | |
|---|---|
| array | Array in which the pointer structure exists. |
| ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |
| linsrt | First element that is moved. |
| mshif | Number of places to be added after *linsrt*. |

### 5.3.2.15 Character Function DPTYPE

Function DPTYPE provides the type of data in the record with *pindex*.

**Calling Sequence:** dptype (array, pindex)

**Data Declaration:** Integer array, pindex

| | | |
|---|---|---|
| **Arguments:** | array | Array in which the pointer structure exists. |
| | pindex | Index of the new pointer. |

#### 5.3.2.16 Integer Function IADRS

Function IADRS provides the address of a record in a pool. If the name of the pointer is not yet present, the index and the address will both be made zero.

**Calling Sequence:** iadrs (array, pindex)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | array, pindex |
| **Arguments:** | array | Array in which pointer structure exists. |
| | pindex | Index of a point. |

#### 5.3.2.17 Integer Function OCINTG

Function OCINTG delivers an integer value stored as a real array.

**Calling Sequence:** ocintg (rvalue)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | rvalue |
| **Arguments:** | rvalue | An integer value. |

#### 5.3.2.18 Real Function OCREAL

Function OCREAL delivers a real value stored in an integer array.

**Calling Sequence:** ocreal (ivalue)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | ivalue |
| **Arguments:** | ivalue | An integer value. |

### *5.3.3 Installation Dependent Subroutines (ocpids FOR Files)*

#### 5.3.3.1 Subroutine CMTOPL

Subroutine CMTOPL converts paper coordinates (*xp*, *yp* in cm) to (HP) plot units.

| | | |
|---|---|---|
| **Calling Sequence:** | cmtopl (xp, yp, ix, iy) | |
| **Data Declaration:** | Integer | lnt, ix, iy |
| | Real | dashl, xp, yp |
| **Arguments:** | lnt | Line type:<br>1-6: Dashed,<br>10: Continuous. |
| | dashl | Dash length. |
| | xp, yp | Paper coordinates. |
| | ix, iy | Integer numbers. |
| **Common Blocks:** | PLPARM(3)<br>PLPARM(4)<br>PLPARM(5)<br>PLPARM(6) | |

### 5.3.3.2 Subroutine DTSTTI

Subroutine DTSTTI transforms time strings into integer time arrays.

| | | |
|---|---|---|
| **Calling Sequence:** | dtstti (iopt, timstr, dttime) | |
| **Data Declaration:** | Integer | iopt, dttime |
| | Character | timstr |
| **Arguments:** | iopt | Option number. |
| | timstr | Time string. |
| | dttime | Time array elements: year, month, day, hour, minute and second. |

### 5.3.3.3 Subroutine DTTIST

Subroutine DTTIST transforms integer time arrays into time strings.

| | | |
|---|---|---|
| **Calling Sequence:** | dttist (iopt, timstr, dttime) | |
| **Data Declaration:** | Integer | iopt, dttime |
| | Character | timstr |
| **Arguments:** | iopt | Option number. |
| | timstr | Time string. |

| | | |
|---|---|---|
| | dttime | Time array elements: year, month, day, hour, minute and second. |

#### 5.3.3.4 Subroutine OCDTIM

Using processor dependent routines, subroutine OCDTIM gets the time of processing.

**Calling Sequence:** ocdtim (prctim)

**Data Declaration:** Integer prctim

| | | |
|---|---|---|
| **Arguments:** | prctim | Time array elements: year, month, day, hour, minute and second. |

#### 5.3.3.5 Subroutine OCPINI

Subroutine OCPINI initializes a number of common variables and opens standard input and output files if necessary.

**Calling Sequence:** ocpini (inifil, lread, inerr)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | inerr |
| | Logical | lread |
| | Character | inifil |

| | | |
|---|---|---|
| **Arguments:** | inerr | Number of the initialization error. |
| | inifil | Name of the initialization file. |
| | lread | If true, command input file must be opened and command reading must be initialized. |

#### 5.3.3.6 Subroutine OPENDF

Subroutine OPENDF terminates a picture.

#### 5.3.3.8 Subroutine OPFRAM

Subroutine OPFRAM plots the edge of the figure and the captions.

**Calling Sequence:** opfram (fropt, ptitl)

**Data Declaration:** Integer fropt

| | | |
|---|---|---|
| | Character | ptitl |
| **Arguments:** | fropt | Frame option:<br>= 0 No frame,<br>= 1 Simple frame,<br>= 2 DUT frame. |
| | ptitl | Figure title. |
| **Common Blocks:** | FILENM<br>XASL<br>YASL<br>SYMSIZ<br>XPLO<br>XPHI<br>YPLO<br>YPHI<br>SUBLNS<br>XPSUB<br>YPSUB | |

#### 5.3.3.9 Subroutine OPINIT

Subroutine OPINIT starts the plotting of a figure and opens the plot file if necessary.

| | | |
|---|---|---|
| **Calling Sequence:** | opinit (xflen, yflen) | |
| **Data Declaration:** | Real | xflen, yflen |
| **Arguments:** | xflen | Length of figure in x-direction. |
| | yflen | Length of figure in y-direction. |
| **Common Blocks:** | FILENM | |

#### 5.3.3.10 Subroutine OPMARK

Subroutine OPMARK plots a single (centered) symbol.

| | | |
|---|---|---|
| **Calling Sequence:** | opmark (xt, yt, syms, isym, updown) | |
| **Data Declaration:** | Integer | isym |
| | Real | xt, yt, syms |
| | Character | updown |

| | | |
|---|---|---|
| **Arguments:** | xt, yt | Place where the first character is plotted. |
| | syms | Size of the symbols on the plot. |
| | isym | Indicator of the symbol to be plotted. Symbol is centered at (*xt, yt*). |
| | updown | = up Pen moves to (*xt, yt*) with pen up;<br>= down Pen moves to (*xt, yt*) with pen down. |

### 5.3.3.11 Subroutine OPNPEN

Subroutine OPNPEN puts on a new plotting pen (with different color).

**Calling Sequence:** opnpen (ipen)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | ipen |

| | | |
|---|---|---|
| **Arguments:** | ipen | Number of the new pen. |

### 5.3.3.12 Subroutine OPPLOT

Subroutine OPPLOT moves the pen to the location (*xt, yt*).

**Calling Sequence:** opplot (xt, yt, updown)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xt, yt |
| | Character | updown |

| | | |
|---|---|---|
| **Arguments:** | xt, yt | Place where the first character is plotted. |
| | updown | = up Pen moves to (*xt, yt*) with pen up;<br>= down Pen moves to (*xt, yt*) with pen down. |

### 5.3.3.13 Subroutine OPTEXT

Subroutine OPTEXT plots a string.

**Calling Sequence:** optext (xt, yt, syms, string, angl, nc)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | nc |
| | Real | xt, yt, syms, angl |
| | Character | string |

| | | |
|---|---|---|
| **Arguments:** | xt, yt | Place where the first character is plotted. |
| | syms | Size of symbols on plot. |

| | |
|---|---|
| strng | Character string to be plotted. |
| angl | Angle under which the string is plotted. |
| nc | Number of characters in the string. |

#### 5.3.3.14 Subroutine OPTYPE

Subroutine OPTYPE plots a new line type.

**Calling Sequence:** optype (lnt, dashl)

**Data Declaration:**

| | |
|---|---|
| Integer | lnt |
| Real | dashl |

**Arguments:**

| | |
|---|---|
| lnt | Line type:<br>1-6: Dashed;<br>10: Continuous. |
| dashl | Dash length. |

### *5.3.4 Plot Routines (ocplot FOR File)*

#### 5.3.4.1 Subroutine ISOLIN

Subroutine ISOLIN computes one contour line, starting from a given point in a given mesh. Modify *idir* (contour direction) if necessary by determining the line on which the next contour point is searched and then determining the first guess of the new point. Call search after the two steps above to determine a new contour point, if a new point is on the edge of the mesh, move to new the mesh.

**Calling Sequence:** isolin (f, cval, fstep, cf, bpost, idir0, ix0, iy0, srx0, sry0, start, pstat, ibx, iby, errc)

**Data Declaration:**

| | |
|---|---|
| Integer | ibx, iby, idir0, ix0, iy0, pstat, start |
| Real | cf, cval, f, fstep, srx0, sry0 |
| Logical | bpost |
| Character | errc |

**Arguments:**

| | |
|---|---|
| ibx | Test for x-connection between neighboring points;<br>*ibx* = 0: no test. |
| iby | Test for y-connection between neighboring points;<br>*iby* = 0: no test. |
| idir0 | Initial direction of contour line *idir0*<br>= 1: -45 <= direction <= 45 degrees; |

| | |
|---|---|
| | = 2: 45 <= direction <= 135 degrees; |
| | = 3: 135 <= direction <= 215 degrees; |
| | = 4: 215 <= direction <= 305 degrees. |
| ix0 | X-coordinate of starting mesh. |
| iy0 | Y-coordinate of starting mesh. |
| pstat | Status in points of grid. |
| start | Indicates whether a new contour line may start in given mesh. |
| cf | Function values are divided by *cf*. |
| cval | Value of function on contour line. |
| f | Values of function to be contoured. |
| fstep | Contour line interval. |
| srx0 | Start point in the mesh, 0 <= *srx0* <= 1. |
| sry0 | Start point in the mesh, 0 <= *sry0* <= 1. |
| bpost | Indicates whether posting of the function value is to be done. |
| errc | Error condition code. |

### 5.3.4.2 Subroutine OCPISO

Subroutine OCPISO organizes the plotting of contour lines. The procedure consists of the following steps:

1) Determine gradients in points where F > 0.
2) Extrapolate where F = 0 (if *cpos* = pos).
3) Start contour lines from boundary points.
4) Start contour lines from interior points.

**Calling Sequence:** ocpiso (cpos, ibx, iby, pstat, f, fmin, fstep, fmax, cf, start)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | ibx, iby, pstat, start |
| | Real | cf, f, fmin, fmax, fstep |
| | Character | cpos |

| | | |
|---|---|---|
| **Arguments:** | ibx | Test for x-connection between neighboring points; *ibx* = 0: no test. |
| | iby | Test for y-connection between neighboring points; *iby* = 0: no test. |
| | pstat | Status in points of the grid. Point status is encoded in array *pstat* as follows:<br>Index im = ixq + (iyq-1)*mxq denotes point (ixq, iyq);<br>If *ibx* and *iby* are zero, it is assumed that all connections exist.<br>Otherwise: |

| | | |
|---|---|---|
| | | If iand(*pstat*(im), *ibx*) = 0, then connection between points (ixq, iyq) and (ixq+1, iyq) is absent. If iand(*pstat*(im), *iby*) = 0, then connection between points (ixq, iyq) and (ixq, iyq+1) is absent. |
| | start | For each mesh indicates:<br>= 0 Contour line went through this mesh;<br>= 1 New contour line can start in this mesh. |
| | cf | Function values appearing on plot are divided by *cf*. |
| | f | Values of function to be contoured. |
| | fmax | Highest contour value. |
| | fmin | Lowest contour value. |
| | fstep | Contour function interval. |
| | cpos | When equal to pos, it means that $f >= 0$. |
| **Common Blocks:** | MXQ<br>MYQ<br>DXQ<br>DYQ | |

### 5.3.4.3 Subroutine OCPSCH

Subroutine OCPSCH determines a scale factor for a plot. The resulting scale *rsc* must be smaller than *slm*, and it must be a number of the form 10**N, 2*10**N, or 5*10**N.

| | | |
|---|---|---|
| **Calling Sequence:** | ocpsch (slm, rsc) | |
| **Data Declaration:** | Real | slm, rsc |
| **Arguments:** | slm | Maximum size of the scale factor. |
| | rsc | Chosen scale factor. |

### 5.3.4.4 Subroutine OCPSUB

Subroutine OCPSUB plots part of the legend under a figure.

| | | |
|---|---|---|
| **Calling Sequence:** | ocpsub (cquan, qsca, qr, qunit) | |
| **Data Declaration:** | Real | qr, qsca |
| | Character | cquan, qunit |
| **Arguments:** | cquan | One of several cases:<br>= delt Function increment is plotted; |

| | | |
|---|---|---|
| | | = lens  A length scale is plotted; |
| | | = arow  A vector scale is plotted; |
| | | = other  The text *cquan* is plotted. |
| | qsca | Length or vector scale: |
| | | Input if *cquan* = lens or arow; |
| | | Output if *cquan* = delt. |
| | qr | Number to be plotted: |
| | | Output if *cquan* = lens or arow; |
| | | Input if *cquan* = delt. |
| | qunit | Unit of the plotted quantity. |
| **Common Blocks:** | XASL | |
| | YASL | |
| | PMR | |
| | SYMSIZ | |

#### 5.3.4.5 Subroutine OCPVEC

Subroutine OCPVEC plots a vector field.

**Calling Sequence:** ocpvec (vsca, vvx, vvy, stag, ibd, pstat, idist)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | ibd, idist, pstat |
| | Real | vsca, vvx, vvy |
| | Logical | pstag, pms |
| | Character | stag |
| **Arguments:** | ibd | If non-zero: tests with *pstat* whether depth is positive or not. |
| | pstat | Encodes the status in points of the grid. |
| | idist | Number of meshes between vector origins. |
| | vsca | Vector scale. |
| | vvx | Array containing x-components of vector. |
| | vvy | Array containing y-components of vector. |
| | pstag | True if staggered grid. |
| | pms | Test for positive depth. |
| | stag | Staggered grid, |
| | | Other: non-staggered grid. |

#### 5.3.4.6 Subroutine OPNUMB

Subroutine OPNUMB plots a real number. The number is converted to a string and then written to a file using subroutine OPTEXT.

| | | |
|---|---|---|
| **Calling Sequence:** | opnumb (xt, yt, syms, reval, angl, ndec) | |
| **Data Declaration:** | Integer | ndec |
| | Real | xt, yt, syms, angl, reval |
| **Arguments:** | xt, yt | Place where the first character is plotted. |
| | syms | Size of the symbols on the plot. |
| | reval | Real number to be plotted. |
| | angl | Angle under which the number is plotted. |
| | ndec | Number of decimals. |

### 5.3.4.7 Subroutine OPSYMB

Subroutine OPSYMB plots a single (centered and oriented) symbol.

| | | |
|---|---|---|
| **Calling Sequence:** | opsymb (xt, yt, syms, isym, angle, updown) | |
| **Data Declaration:** | Integer | isym |
| | Real | xt, yt, syms, angle |
| | Character | updown |
| **Arguments:** | isym | Indicator of the symbol to be plotted. Symbol is centered at (*xt, yt*). |
| | syms | Size of the symbols on the plot. |
| | xt, yt | Place where the first character is plotted. |
| | angle | Angle under which the symbol must be plotted. |
| | updown | = up Pen moves to (*xt, yt*) with pen up;<br>= down Pen moves to (*xt, yt*) with pen down. |

### 5.3.4.8 Subroutine PLOTF

Subroutine PLOTF plots a point given in window (physical) coordinates.

| | | |
|---|---|---|
| **Calling Sequence:** | plotf (xf, yf, updown) | |
| **Data Declaration:** | Real | xf, yf |
| | Character | updown |
| **Arguments:** | xf, yf | Window coordinates. |
| | updown | Pen up or down when moving to the point. |

### 5.3.4.9 Subroutine PSYM

**Calling Sequence:** psym (xf, yf, syms, isym, updown)

**Data Declaration:**

| | |
|---|---|
| Real | xf, yf, syms |
| Character | updown |
| Integer | isym |

**Arguments:**

| | |
|---|---|
| xf, yf | Place whereto the pen must move and where the symbol must appear in paper coordinates (cm). |
| syms | Size of symbols on plot (cm). |
| isym | Symbol indicator. |
| updown | Pen up or down when moving to the point. |

### 5.3.4.10 Subroutine SNYPT1

Subroutine SNYPT1 determines the crossing point of a line segment with the edge of the frame; (*xs, ys*) is the crossing point in paper coordinate (cm). The end points of the line segment are (*x1, y1*) and (*x2, y2*). It is assumed that (*x1, y1*) is inside the frame, and (*x2, y2*) outside.

**Calling Sequence:** snypt1 (x1, y1, x2, y2, xs, ys)

**Data Declaration:** Real x1, y1, x2, y2, xs, ys

**Arguments:**

| | |
|---|---|
| x1 | X of the begin point. |
| y1 | Y of the begin point. |
| x2 | X of the end point. |
| y2 | Y of the end point. |
| xs | X of the crossing. |
| ys | Y of the crossing. |

### 5.3.4.11 Subroutine SNYPT2

Subroutine SNYPT2 determines the number of crossing points and their coordinates of a line segment with the plotting frame. Both ends of the line segment should be outside the plotting frame. First check whether the line segment lies fully right, left, top or bottom of the plotting frame. When this is not the case it looks for possible cross-sections with all four sides of the plotting frame.

**Calling Sequence:** snypt2 (x1, y1, x2, y2, xs1, ys1, xs2, ys2, nsnypt)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | nsnypt |
| | Real | s1, y1, x2, y2, xs1, ys1, xs2, ys2 |
| **Arguments:** | nsnypt | Total number of crossing points. |
| | xs1 | X-coordinate of the first cross-section. |
| | xs2 | X-coordinate of the second cross-section. |
| | x1 | X-coordinate of the begin line segment. |
| | x2 | X-coordinate of the end line segment. |
| | ys1 | Y-coordinate of the first cross-section. |
| | ys2 | Y-coordinate of the second cross-section. |
| | y1 | Y-coordinate of the begin line segment. |
| | y2 | Y-coordinate of the end line segment. |
| **Common Blocks:** | OUTPDA | |

### *5.3.5 Miscellaneous Routines (ocpmix FOR Files)*

#### 5.3.5.1 Subroutine BUGFIX

Subroutine BUGFIX adds one character to the version character string.

| | | |
|---|---|---|
| **Calling Sequence:** | bugfix (fixabc) | |
| **Data Declaration:** | Character | fixabc |
| **Arguments:** | fixabc | Character indicating a bugfix. |

#### 5.3.5.2 Subroutine DTINTI

Subroutine DTINTI calculates integer time array *inttim* from time in seconds for a given reference day *refday*. Every fourth year is a leap year except century-years. Leap years also include year 0, 1000, 2000 etc. The first day of January of year zero is day number one.

| | | |
|---|---|---|
| **Calling Sequence:** | dtinti (timesc, inttim) | |
| **Data Declaration:** | Integer | inttim |
| | Real | timesc |
| **Arguments:** | inttim | (1) Year; |
| | | (2) Month; |
| | | (3) Day; |

| | | |
|---|---|---|
| | | (4) Hour; |
| | | (5) Minute; |
| | | (6) Second. |
| | timesc | Time in seconds from given reference day refday. |

#### 5.3.5.3 Subroutine DTRETI

**Calling Sequence:** dtreti (tstrng, iopt, timesc)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | iopt |
| | Real | timesc |
| | Character | tstrng |

| | | |
|---|---|---|
| **Arguments:** | iopt | Option number. |
| | timesc | Time in seconds from given reference day refday. |
| | tstrng | Time string. |

#### 5.3.5.4 Real Function DTTIME

Function DTTIME gives the time in seconds from a reference day. It also initializes the reference day. Every fourth year is a leap year except century-years. Leap years also include year 0, 1000, 2000 etc. The first of January of year zero is day number one.

**Calling Sequence:** dttime (inttim)

**Data Declaration:** Integer inttim

| | | |
|---|---|---|
| **Arguments:** | inttim | (1) Year; |
| | | (2) Month; |
| | | (3) Day; |
| | | (4) Hour; |
| | | (5) Minute; |
| | | (6) Second. |

**Common Blocks:** REFDAY

#### 5.3.5.5 Character Function DTTIWR

**Calling Sequence:** dttiwr (iopt, timesc)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | iopt |
| | Real | timesc |

| | | |
|---|---|---|
| | Character | tstrng |
| **Arguments:** | iopt | Time coding option number. |
| | timesc | Time in seconds from given reference day refday. |
| | tstrng | Time string. |

### 5.3.5.6 Logical Function EQREAL

Function EQREAL determines whether a value (usually a value read from file) is an exception value or not. Function EQREAL is later used to make comparisons of floating points within reasonable bounds.

**Calling Sequence:** eqreal (real1, real2)

| | | |
|---|---|---|
| **Data Declaration:** | Real | real1, real2 |
| **Arguments:** | real1 | Value that is to be tested. |
| | real2 | The given exception value. |

### 5.3.5.7 Subroutine FOR

Subroutine FOR is a general open file routine.

**Calling Sequence:** for (iunit, ddname, sf, iostat)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | iunit, iostat |
| | Character | ddname, sf |
| **Arguments:** | iunit | = 0 Get free unit number;<br>> 0 Fixed unit number;<br>Output: allocated unit number. |
| | ddname | Filename string (empty if *iunit* > 0). |
| | sf | File qualifiers:<br>1$^{st}$ character: O(ld), N(ew), S(cratch), U(nknown);<br>2$^{nd}$ character: F(ormatted), U(nformatted). |
| | iostat | = 0 Full messages printed;<br>= -1 Only error messages printed;<br>= -2 No messages printed;<br>Output: error indicator. |

#### 5.3.5.8 Subroutine INAR2D

Subroutine INAR2D reads a 2-D array from a data set and is used to read bathymetry, one component of wind velocity.

| | | |
|---|---|---|
| **Calling Sequence:** | inar2d (arr, mxa, mya, ndsl, ndsd, idfm, rform, idla, vfac, nhed, nhedf) | |
| **Data Declaration:** | Integer | idfm, idla, mxa, mya, ndsd, ndsl, nhed, nhedf |
| | Real | arr, vfac |
| | Character | rform |
| **Arguments:** | idfm | Format index. |
| | idla | Layout indicator. |
| | mxa | Number of points along x-side of grid. |
| | mya | Number of points along y-side of grid. |
| | ndsd | Unit number of the file from which to read the data set. |
| | ndsl | Unit number of the file containing the list of filenames. |
| | nhedf | Number of heading lines in the file (first lines). |
| | nhedl | Number of heading lines in the file before each array. |
| | arr | Results appear in this array. |
| | rform | Format used in reading data (character string). |
| | vfac | Factor by which data must be multiplied. |

#### 5.3.5.9 Subroutine LSPLIT

Subroutine LSPLIT separates a line read from a file into single data items. Each data item is found in a string *datitm.*

| | | |
|---|---|---|
| **Calling Sequence:** | lsplit (reline, datitm, numitm) | |
| **Data Declaration:** | Integer | numitm |
| | Character | reline, datitm |
| **Arguments:** | numitm | Maximum number of data items in the array. |
| | datitm | Array of data items. |
| | reline | String (read from an input file). |

#### 5.3.5.10 Subroutine MSGERR

Subroutine MSGERR produces error messages. If necessary, the value of leverr is increased. In case of a high error level an error message file is opened.

**Calling Sequence:** msgerr (lev, string)

**Data Declaration:**

| | |
|---|---|
| Integer | lev |
| Character | string |

**Arguments:**

| | |
|---|---|
| lev | Indicates how severe the present error is. |
| string | Contents of the present error message. |

#### 5.3.5.11 Subroutine REPARM

Subroutine REPARM reads parameters used for reading an array from user input.

**Calling Sequence:** reparm (ndsl, ndsd, idla, idfm, rform, nhedf, logt, nhedt, logc, nhedc)

**Data Declaration:**

| | |
|---|---|
| Integer | idfm, idla, ndsl, ndsd, nhedf, nhedt, nhedc |
| Logical | logt, logc |
| Character | rform |

**Arguments:**

| | |
|---|---|
| idfm | Format index. |
| idla | Layout indicator. |
| ndsd | Unit number of the file from which to read the data set. |
| ndsl | Unit number of the file containing the list of filenames. |
| nhedf | Number of heading lines in the file (once in each file). |
| nhedt | Number of heading lines in the file before reading each time level. |
| nhedc | Number of heading lines in the file before each array or vector component. |
| logt | If true, then the field is time-dependent. |
| logc | If true, then more than one component is read from the file. |
| rform | Reading format. |

### 5.3.5.12 Logical Function STPNOW

Function STPNOW determines whether the SWAN program should be stopped due to a terminating error. STPNOW compares two common variables. The maximum allowable error-level, maxerr, and the actual error-level, leverr.

### 5.3.5.13 Subroutine STRACE

Subroutine STRACE produces, depending on the value of itrace, a message containing the name *subnam*. The purpose of this action is to detect the entry of a subroutine. The first executable statement of subroutine AAA (which is a name for any subroutine) must be: CALL STRACE(IENT, AAA). Further if necessary: DATA IENT/0/ If ITRACE = 0, no message. If ITRACE > 0, a message is printed up to itrace times.

| | | |
|---|---|---|
| **Calling Sequence:** | strace (ient, subnam) | |
| **Data Declaration:** | Integer | ient |
| | Character | subnam |
| **Arguments:** | ient | Number of entries into the calling subroutine. |
| | subnam | Name of the calling subroutine. |

### 5.3.5.14 Subroutine TABHED

Subroutine TABHED prints the table heading that contains the run description, three lines, name of institute, program name, project name, and run ID.

| | | |
|---|---|---|
| **Calling Sequence:** | tabhed (prognm, lpr) | |
| **Data Declaration:** | Integer | lpr |
| | Character | prognm |
| **Arguments:** | lpr | Unit reference number. |
| | prognm | Program name. |

### *5.3.6 Computation Subroutines (swancom1 FOR File)*

#### 5.3.6.1 Subroutine ACTION

Subroutine ACTION determines the transportation, refraction and source terms of the ACTION balance equation.

**Calling Sequence:** action (idcmin, idcmax, spcsig, ac2, cax, cay, cas, cad, imatla, imatda, imatua, imatra, warea, sector, imat5l, imat6u, iscmin, iscmax, iddlow, iddtop, isstop, anyblk, anybin, leakc1, ac1, dyndep, rdx, rdy, swpdir, ix, iy, ksx, ksy, obsta, xcgrid, ycgrid, cross, iter, kgrpnt, dep2, chs, obredf, wlev2, cax1, cay1, spcdir, cgo)

**Data Declaration:**

| | |
|---|---|
| Real | spcsig, xcgrid, ycgrid, ac2, cax, cay, cax1, cay2, cgo, cas, cad, imatla, imatda, imatua, imatra, imat5l, imat6u, leakc1, rdx, rdy, dep2, obredf, wlev2, chs, spcdir, ksx, ksy |
| Integer | warea, idcmin, idcmax, iscmin, iscmax, sector, obsta, kgrpnt, cross, iddlow, iddtop, isstop, swpdir, iter, ix, iy, supdir |
| Logical | anyblk, anybin, dyndep |

**Arguments:**

| | |
|---|---|
| spcsig | Relative frequencies in computational domain in sigma space. |
| xcgrid | X-coordinate of computational grid in x-direction. |
| ycgrid | Y-coordinate of computational grid in y-direction. |
| idcmin | Integer array containing minimum counter. |
| idcmax | Integer array containing maximum counter. |
| ac2 | Action density as function of D, S, X, Y at time T. |
| cax | Wave transport velocity in x-direction as function of (*id, is, ic*). |
| cay | Wave transport velocity in y-direction as function of (*id, is, ic*). |
| cas | Wave transport velocity in frequency-direction as function of (*id, is, ic*). |
| cad | Wave transport velocity in spectral direction as function of (*id, is, ic*). |
| imatla | Coefficients of lower diagonal of matrix. |
| imatda | Coefficients of diagonal of matrix. |
| imatua | Coefficients of upper diagonal of matrix. |
| imatra | Coefficients of right-hand side of matrix. |
| warea | The big array used in data pool scheme, to contain many variables. |

| | |
|---|---|
| sector | Indicates which configuration is present. |
| imat5l | Coefficient of lower diagonal in the presence of a current. |
| imat6u | Coefficient of upper diagonal in the presence of a current. |
| iscmin | Frequency dependent counter in frequency space. |
| iscmax | Frequency dependent counter in frequency space. |
| iddlow | Minimum counter per sweep taken over all frequencies. |
| iddtop | Maximum counter per sweep taken over all frequencies. |
| isstop | Maximum frequency counter for wave components that are propagated within a sweep. |
| anyblk | 2D Determines if a bin is BLOCKED by a counter current based on a CFL criterion. |
| anybin | = True, if a certain bin is enclosed in a sweep. |
| leakc1 | Leak coefficient. |
| ac1 | Action density as function of D, S, X, Y at time T. |
| dyndep | If true, depths vary with time. |
| rdx, rdy | Array containing spatial derivative coefficient. |
| swpdir | Current sweep direction. |
| ix | Counter of grid points in x-direction. |
| iy | Counter of grid points in y-direction. |
| ksx | Dummy variable to get the right sign in the numerical difference scheme in x-direction depending on the sweep direction, KSX = ñ1. |
| ksy | Dummy variable to get the right sign in the numerical difference scheme in y-direction depending on the sweep direction, KSY = ñ1. |
| obsta | Array of obstacle parameters. |
| xcgrid | X-coordinate of computational grid in x-direction. |
| ycgrid | Y-coordinate of computational grid in y-direction. |
| cross | Array which contains 0's if there is no obstacle crossing if an obstacle is crossing between the central point and its neighbor *cross* is equal to the number of the obstacle. |
| iter | Iteration counter for SWAN. |
| kgrpnt | Grid point addresses. |
| dep2 | Depth. |
| chs | Sign. wave height in whole computational grid. |
| obredf | Array of action density reduction coefficients. |
| wlev2 | Water level in grid points. |
| cax1 | Propagation velocity in x old time level. |
| cay1 | Propagation velocity in y old time level. |

| | |
|---|---|
| spcdir | (*,1) Spectral directions (radians); |
| | (*,2) Cosine of spectral directions; |
| | (*,3) Sine of spectral directions; |
| | (*,4) Cosine^2 of spectral directions; |
| | (*,5) Cosine*sine of spectral directions; |
| | (*,6) Sine^2 of spectral directions. |
| cgo | Group velocity. |

#### 5.3.6.2 Subroutine INSAC

Subroutine INSAC checks the accuracy of the final computation. If a certain accuracy has been reached it stops the iteration.

**Calling Sequence:** insac (ac2, spcsig, dep2, hsacc2, sacc2)

**Data Declaration:** Real spcsig, ac2, dep2, hsacc2, sacc2

**Arguments:**

| | |
|---|---|
| spcsig | Relative frequencies in computational domain in sigma space. |
| dep2 | Depth. |
| ac2 | Action density as function of D, S, X, Y at time T. |
| hsacc2 | Dummy array for the significant wave height (old value). |
| sacc2 | Dummy array for the mean frequency (old value). |

#### 5.3.6.3 Subroutine PHILIM

Subroutine PHILIM limits the change in action density between two iterations to a certain percentage of the Phillips equilibrium level.

**Calling Sequence:** philim (ac2, ac2old, cgo, kwave, spcsig, anybin, qb_loc)

**Data Declaration:**

| | |
|---|---|
| Logical | anybin |
| Real | ac2, ac2old, cgo, kwave, spcsib, qb_lo |

**Arguments:**

| | |
|---|---|
| qb_loc | Local value of *qb* (fraction of breaking waves). |
| ac2 | (Non-stationary case) action density as function of D, S, X, Y at time T + DT. |
| ac2old | Values of action density stored for limiter. |
| cgo | Group velocity. |
| kwave | Wave number as function of the relative frequency S and position ic (*ix, iy*). |
| spcsig | Relative frequencies in computational domain in |

| | |
|---|---|
| | sigma space. |
| anybin | = True if a certain bin is enclosed in a sweep. Array is used to determine whether or not some coefficients in the array have to be changed. |

### 5.3.6.4 Subroutine RESCALE

Subroutine RESCALE removes negative values from a computed action density spectrum.

**Calling Sequence:** rescale (ac2, isstop, idcmin, idcmax)

| | | |
|---|---|---|
| **Data Declaration:** | Real | ac2 |
| | Integer | idcmin, idcmax, isstop |
| **Arguments:** | ac2 | Action densities. |
| | isstop | Maximum frequency counter in this sweep. |
| | idcmin | Integer array containing minimum counter of directions. |
| | idcmax | Integer array containing maximum counter. |

### 5.3.6.5 Subroutine SACCUR

Subroutine SACCUR checks the accuracy of the final computation. If a particular accuracy has been reached then the iteration process terminates.

**Calling Sequence:** saccur (dep2, ac2, spcsig, accur, hsacc1, hsacc2, sacc1, sacc2, delhs, deltm)

| | | |
|---|---|---|
| **Data Declaration:** | Real | spcsig, ac2, dep2, hsacc1, hsacc2, sacc1, sacc2, delhs, deltm, accur |
| **Arguments:** | spcsig | Relative frequencies in computational domain in sigma space. |
| | dep2 | Depth. |
| | ac2 | Action density as function of D, S, X, Y at time T. |
| | accur | User specified option used to influence the criterion for terminating the iterative procedure in the SWAN computations. |
| | hsacc1 | Dummy array for the significant wave height (new value). |
| | hsacc2 | Dummy array for the significant wave height (old value). |

| | |
|---|---|
| sacc1 | Dummy array for the mean frequency (new value). |
| sacc2 | Dummy array for the mean frequency (old value). |
| delhs | Difference in Hs between last two iterations. |
| deltm | Difference in Tm between last two iterations. |

### 5.3.6.6 Subroutine SCOMPU

Subroutine SCOMPU is the main subroutine of the computational part.

### 5.3.6.7 Subroutine SINTGRL

Subroutine SINTGRL computes several integrals used in SWAN and some general parameters.

**Calling Sequence:** sintgrl (spcdir, kwave, ac2, dep2, qb_loc, ursell, rdx, rdy, ac2tot, etot, abrbot, ubot, hs, qb, hm, kmespc, smebrk)

**Data Declaration:** Real dep2, kwave, rdx, rdy, spcdir, ac2, qb, ubot, ursell, abrbot, etot, hm, hs, qb_loc, ac2tot, kmespc, smebrk, ac2tot

**Arguments:**

| | |
|---|---|
| abrbot | Near bottom excursion. |
| ac2 | Action density as a function of *id, is, ix* and *iy*. |
| ac2tot | Total action density per grid point. |
| dep2 | Water depth. |
| etot | Total wave energy density. |
| hm | Maximum wave height. |
| hs | Significant wave height. |
| kmespc | Mean average wave number according to the WAM formulation. |
| kwave | Wave number function of frequency and ic. |
| qb | Fraction of breaking waves. |
| qb_loc | Fraction of breaking waves at current grid point. |
| smebrk | Mean frequency according to first order moment. |
| ubot | Near bottom velocity as function of *ix* and *iy*. |
| ursell | *Ursell* number as function of *ix* and *iy*. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| rdx, rdy | Array containing spatial derivative coefficient. |

#### 5.3.6.8 Subroutine SOLBAND

Subroutine SOLBAND solves the array in the case of a current. A fully implicit scheme in frequency and directional space is used. Dr. C. Vuik, from Delft University of Technology in the Netherlands, has provided the subroutines that solve this matrix.

**Calling Sequence:** solband (band, exact, rhv, rinsol, solut, work, precon, upperi, loperi, anybin, infmat, iinsol, imatra, imatla, imatda, imatua, imat5l, imat6u, ac2old, cgo, kwave, spcsig, idcmin, idcmax, ac2, sector, iter, idtot, istot, iddlow, iddtop, isstop, inocnv, qbloc, errpts, ix, iy, itsw)

**Data Declaration:**

| | |
|---|---|
| Integer | iter, itsw, iddlow, inocnv, iddtop, idtot, istot, isstop, errpts, infmat, iinsol, idcmin, idcmax, sector |
| Real | spcsig, exact, rhv, solut, work, precon, imatra, imatla, imatda, imatua, imat5l, imat6u, ac2old, cgo, idcmin, idcmax, ac2, qbloc, rinsol, upperi, loperi, kwave |
| Logical | anybin |

**Arguments:**

| | |
|---|---|
| iter | Iteration counter for SWAN. |
| itsw | Timestep counter for SWAN. |
| spcsig | Relative frequencies in computational domain in sigmaspace. |
| ix | Counter of grid points in x-direction. |
| iy | Counter of grid points in y-direction. |
| idtot, istot | Maximum range between the minimum and maximum counter in directional and frequency space, respectively. |
| band | Matrix from the equations to be solved. |
| exact | Exact Solution. |
| rhv | Right-hand side. |
| rinsol | Real information for the solver. |
| solut | Iterative solution. |
| work | Work space. |
| precon | Preconditioner. |
| upperi | Only relevant for computation in periodic domain. |
| loperi | Only relevant for computation in periodic domain. |
| anybin | = True if a certain bin is enclosed in a sweep. |
| infmat | Integer information for the matrix. |
| iinsol | Integer information for the solver. |
| imatda | Coefficients of diagonal of the matrix. |
| imatla | Coefficients of lower diagonal of the matrix. |

| | |
|---|---|
| imatua | Coefficients of upper diagonal of the matrix. |
| imatra | Coefficients of right-hand side of the matrix. |
| imat5l | Coefficients for implicit calculation in frequency space (lower diagonal). |
| imat6u | Coefficients for implicit calculation in frequency space (upper diagonal). |
| ac2old | Values of action density stored for limiter. |
| cgo | Group velocity. |
| kwave | Wave number as function of the relative frequency S and position ic (*ix, iy*). |
| idcmin | Integer array containing minimum counter. |
| idcmax | Integer array containing maximum counter. |
| ac2 | Action density as function of D, S, X, Y and T. |
| sector | Indicates which configuration is present. |
| iddlow | Minimum counter per sweep taken over all frequencies. |
| iddtop | Maximum counter per sweep taken over all frequencies. |
| isstop | Maximum frequency counter for wave components that are propagated within a sweep. |
| inocnv | Counts occurrence of nonconvergence in solver. |
| qbloc | Fraction of breaking waves at current grid point. |
| errpts | Info for SWAN to keep track of grid points (x,y) at which errors occur. |

### 5.3.6.9 Subroutine SOLMAT

Subroutine SOLMAT solves the matrix that is filled in subroutine ACTION. The solutions give the values for the wave action for every frequency and every direction. Only the Thomas Sweep Algorithm in the spectral direction solves the matrices.

**Calling Sequence:** solmat (idcmin, idcmax, ac2, imatra, imatda, imatua, imatla, ac2old, kwave, cgo, spcsig, qbloc)

**Data Declaration:**

| | |
|---|---|
| Real | spcsig, qbloc, ac2, imatda, imatla, imatua, imatra, ac2old, kwave |
| Integer | idcmin, idcmax |

**Arguments:**

| | |
|---|---|
| spcsig | Relative frequencies in computational domain in sigma space. |
| idcmin | Integer array containing minimum counter. |
| idcmax | Integer array containing maximum counter. |
| ac2 | Action density as a function of D, S, X, Y and T. |
| imatda | Coefficients of a diagonal of matrix. |

| | |
|---|---|
| imatla | Coefficients of lower diagonal of matrix. |
| imatua | Coefficients of upper diagonal of matrix. |
| imatra | Coefficients of right-hand side of matrix. |
| ac2old | Values of action density stored for limiter. |
| kwave | Wave number as function of the relative frequency S and position ic (*ix, iy*). |
| cgo | Group velocity. |
| qbloc | Fraction of breaking waves at current grid point. |

### 5.3.6.10 Subroutine SOLMT1

Subroutine SOLMT1 solves the matrix that is filled in subroutine ACTION. The solutions give the values for the wave action for every frequency and direction. Only the Thomas Sweep Algorithm in the spectral direction solves the matrices.

**Calling Sequence:** solmt1 (idcmin, idcmax, ac2, imatra, imatda, imatua, imatla, ac2old, kwave, cgo, spcsig, sector, icolu2, anybin, qbloc, isstop, anyblk, iddlow, iddtop

**Data Declaration:**

| | |
|---|---|
| Real | ac2, imatra, imatda, imatua, imatla, ac2old, kwave, cgo, spcsig, qbloc, icolu2 |
| Integer | idcmin, idcmax, sector, isstop, iddtop, iddlow |
| Logical | anybin, anyblk |

**Arguments:**

| | |
|---|---|
| spcsig | Relative frequencies in computational domain in sigma space. |
| ac2 | Action density as function of D, S, X, Y and T. |
| imatda | Coefficients of the diagonal of the matrix. |
| imatla | Coefficients of the lower diagonal of the matrix. |
| imatua | Coefficients of the upper diagonal of the matrix. |
| imatra | Coefficients of the right-hand side of the matrix. |
| cgo | Group velocity. |
| idcmin | Integer array containing minimum counter. |
| idcmax | Integer array containing maximum counter. |
| sector | Sectors enclosed in a sweep. |
| anybin | = True if a certain bin is enclosed in a sweep. The array is used to determine whether some coefficients in the array must be changed. |
| icolu2 | Auxiliary array for storing the coefficients in the last column. |
| kwave | Wave number as a function of the relative frequency S and position ic (*ix, iy*). |
| qbloc | Fraction of breaking waves at current grid point. |
| isstop | Maximum frequency counter for wave components |

| | |
|---|---|
| | that are propagated within a sweep. |
| anyblk | Determines if a bin is BLOCKED by a counter current based on a CFL criterion. |
| iddlow | Minimum counter per sweep taken over all frequencies. |
| iddtop | Maximum counter per sweep taken over all frequencies. |

### 5.3.6.11 Subroutine SOLPRE

Subroutine SOLPRE copies local spectrum to array *ac2old*, and writes the test output fill array for non-active bins.

**Calling Sequence:** solpre (ac2, ac2old, imatra, imatla, imatda, imatua, imat5l, imat6u, idcmin, idcmax, sector, anybin, idtot, istot, iddlow, iddtop, isstop, inocnv)

**Data Declaration:**

| | |
|---|---|
| Real | ac2, ac2old, imatda, imatla, imatua, imatra, imat5l, imat6u |
| Integer | idcmin, idcmax, iddlow, inocnv, iddtop, idtot, istot, isstop, sector |
| Logical | anybin |

**Arguments:**

| | |
|---|---|
| ac2 | Action density as function of D, S, X, Y and T. |
| ac2old | Values of action density stored for limiter. |
| imatda | Coefficients of diagonal of the matrix. |
| imatla | Coefficients of lower diagonal of the matrix. |
| imatua | Coefficients of upper diagonal of the matrix. |
| imatra | Coefficients of right-hand side of the matrix. |
| imat5l | Coefficients for implicit calculation in frequency space (lower diagonal). |
| imat6u | Coefficients for implicit calculation in frequency space (upper diagonal). |
| idcmin | Integer array containing minimum counter. |
| idcmax | Integer array containing maximum counter. |
| sector | Indicates which configuration is present. |
| anybin | = True if a certain bin is enclosed in a sweep. |
| idtot, istot | Maximum range between the minimum and maximum counter in directional and frequency space, respectively. |
| iddlow | Minimum counter per sweep taken over all frequencies. |
| iddtop | Maximum counter per sweep taken over all frequencies. |

| | |
|---|---|
| isstop | Maximum frequency counter for wave components that are propagated within a sweep. |
| inocnv | Counts occurrence of nonconvergence in solver. |

### 5.3.6.12 Subroutine SOURCE

Subroutine SOURCE computes the source terms, i.e., bottom friction, wave breaking, wind input, whitecapping and non-linear wave-wave interactions.

**Calling Sequence:** source (iter, ix, iy, swpdir, kwave, spcsig, ecos, esin, ac2, dep2, imatda, imatra, abrbot, kmespc, smespc, ubot, ufric, ux2, uy2, idcmin, idcmax, iddlow, iddtop, idwmin, idwmax, isstop, plwnda, plwndb, plwcap plbtfr, plwbrk, plnl4s, plnl4d, pltri, warea, hs, etot, qbloc, thetaw, hm, fpm, wind10, etotw, groww, alimw, smebrk, snlc1, fachfr, dal1, dal2, dal3, af11, ue, sa1, sa2, da1c, da1p, da1m, da2c, da2p, da2m, sfnl, dsnl, memnl4, wwint, wwawg, wwswg, cgo, ustar, zelen, spcdir, anywnd, dissc0, dissc1, szeroc, eps2wc, diswcp, wcpsme, wcpkme, wcpqb, wcphm, xis, frcoef, it, precor, ursell)

**Data Declaration:**

| | |
|---|---|
| Real | ecos, esin, spcdir, spcsig, abrbot, etot, hm, qbloc, etotw, fpm, wind10, thetaw, smespc, kmespc, snlc1, fachfr, dal1, dal2, dal3, ufric, smebrk, hs, szeroc, eps2wc, diswcp, wcpqb, wcphm, wcpsme, wcpkme, xis, ac2, dep2, alimw, imatda, imatra, kwave, ubot, ux2, uy2, af11, ue, sa1, sa2, da1c, da1p, da1m, da2c, da2p, da2m, sfnl, dsnl, memnl4, plwnda, plwndb, plwcap, plbtfr, plwbrk, plnl4s, plnl4d, pltri, wwawg, wwswg, cgo, ustar, zelen, dissc0, dissc1, ursell, frcoef, etotw, swpdir |
| Integer | iter, idwmin, idwmax, isstop, iddtop, iddlow, ix, iy, warea, idcmin, idcmax, wwint, it |
| Logical | precor, groww, anywnd |

**Arguments:**

| | |
|---|---|
| ecos | = *spcdir*(*,2) Cosine of spectral directions. |
| esin | = *spcdir*(*,3) Sine of spectral directions. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| spcsig | Relative frequencies in computational domain in sigma space. |

| | |
|---|---|
| iter | Iteration counter for SWAN. |
| ix | Counter of grid points in x-direction. |
| iy | Counter of grid points in y-direction. |
| swpdir | Current sweep direction. |
| kwave | Wave number as function of the relative frequency S and position ic (*ix, iy*). |
| ac2 | (Non-stationary case) action density as function of D, S, X, Y at time T + DT. |
| dep2 | (Non-stationary case) depth as a function of X and Y at time T + DIT. |
| imatda | Coefficients of diagonal of matrix. |
| imatra | Coefficients of right-hand side of matrix. |
| abrbot | Near bottom excursion. |
| kmespc | Mean average wave number according to the WAM formulation. |
| smespc | Mean average frequency over full spectrum. |
| ubot | Absolute orbital velocity in a grid point (*ix, iy*). |
| ufric | Wind friction velocity. |
| ux2 | (Non-stationary case) X-component of current velocity in (X, Y) at time T + DIT. |
| uy2 | (Non-stationary case) Y-component of current velocity in (X, Y) at time T + DIT. |
| idcmin | Minimum frequency dependent counter in directional space. |
| idcmax | Maximum frequency dependent counter in directional space. |
| iddlow | Minimum counter per sweep taken over all frequencies. |
| iddtop | Maximum counter per sweep taken over all frequencies. |
| idwmin | Minimum counter for spectral wind direction. |
| idwmax | Maximum counter for spectral wind direction. |
| isstop | Maximum frequency that is propagated within a sweep. |
| plwnda | Values of source term for test point. |
| plwndb | Values of source term for test point. |
| plwcap | Array containing the whitecapping source term for test output. |
| plbtfr | Bottom friction source term array for outputting on one of the source terms at a particular grid point. |
| plwbrk | Surf breaking source term array for outputting on one of the source terms at a particular grid point . |
| plnl4s | Nonlinear source term array (rhs part) for outputting on one of the source terms at a particular grid point. |
| plnl4d | Nonlinear source term array (diagonal part) for |

| | |
|---|---|
| | outputting on one of the source terms at a particular grid point. |
| pltri | Values of the triad source terms in test points. |
| warea | The big array used in data pool scheme, to contain many variables. |
| hs | Significant wave height. |
| etot | Total energy density per grid point. |
| qbloc | Fraction of breaking waves. |
| thetaw | Mean direction of the relative wind vector. |
| hm | Maximum wave height. |
| fpm | PM frequency. |
| wind10 | Velocity of the relative wind vector. |
| etotw | Total energy of the wind sea spectrum. |
| groww | Check for a certain frequency if the waves are growing or not in a spectral direction. |
| alimw | Maximum energy by wind growth. |
| smebrk | Mean frequency according to first order moment. |
| snlc1 | Coefficient for the subroutines SWSNLN. |
| fachfr | Contribution of high frequency tail to wave stress. |
| dal1, dal2, dal3 | Lambda dependent weight factors |
| af11 | Scaling frequency. |
| ue | "Unfolded" spectrum. |
| sa1, sa2 | Interaction contribution of first and second quadrants, respectively (unfolded space). |
| da1c, da1p, da1m, da2c, da2p, da2m | Items for diagonal matrix. |
| sfnl | Source term Snl, rhs part. |
| dsnl | Source term Snl, diag part. |
| memnl4 | Saves sfnl at every x, y point in memory. |
| wwint | Counters for four wave-wave interactions. |
| wwawg | Weight coefficients for the four wave-wave interactions. |
| wwswg | Weights coefficients for the four wave-wave interactions for the semi-implicit computation. |
| cgo | Group velocity. |
| ustar | Friction velocity at previous iteration for Janssen (1989, 1991) wind input formulation. |
| zelen | Roughness length at previous iteration for Janssen (1989, 1991) wind input formulation. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions; |

| | |
|---|---|
| | (*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| anywnd | Indicator if wind input has to be taken into account for a bin. |
| dissc0 | (Not used); Stores the dissipation distributed over spectral space in one point of the computational grid (old value). |
| dissc1 | (Not used); Dissipation coefficient, function of sigma and theta. |
| szeroc | Not used. |
| eps2wc | Not used. |
| diswcp | Not used. |
| wcpsme | Not used. |
| wcpkme | Not used. |
| wcpqb | Not used. |
| wcphm | Not used. |
| xis | Difference between succeeding frequencies. |
| frcoef | Spatially variable friction coefficient. |
| it | Timestep counter for SWAN. |
| precor | Determines whether first guess estimate for stationary mode is calculated. |
| ursell | *Ursell* number as function of *ix* and *iy*. |

#### 5.3.6.13 Subroutine SWCOMP

Subroutine SWCOMP is the main subroutine for the computational module. In subroutine SCOMPU the main processes taking place in the shallow water zone are determined in several subroutines. The input for this subroutine comes from SWANPRE1, SWANPRE2 and SWANPRE3. The output is sent to the subroutines SWANOUT1, SWANOUT2 and SWANOUT3. The output consists of some characteristic wave parameters and the wave action density. The equations are all based on the action density N, which is a function of the spatial position (x, y), the relative frequency(s) and the spectral direction(d).

**Calling Sequence:** swcomp (warea, rwarea, lwarea, ac1, ac2, compda, spcdir, spcsig, swtsda, xytst, it, kgrpnt, xcgrid, ycgrid, obsta, cross)

**Data Declaration:**

| | |
|---|---|
| Real | rwarea, spcdir, spcsig, xcgrid, ycgrid, ac2, ac1, compda, swtsda |
| Logical | lwarea |
| Integer | it, warea, xytst, kgrpnt, obsta, cross |

**Arguments:**

| | |
|---|---|
| rwarea | Real equivalence of *warea*. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions; |

| | |
|---|---|
| | (*,3) Sine of spectral directions; |
| | (*,4) Cosine^2 of spectral directions; |
| | (*,5) Cosine*sine of spectral directions; |
| | (*,6) Sine^2 of spectral directions. |
| spcsig | Relative frequencies in the computational domain in sigma space. |
| xcgrid | X-coordinate of computational grid in x-direction. |
| ycgrid | Y-coordinate of computational grid in y-direction. |
| ac1 | Action density as function of D, S, X, Y at time T. |
| ac2 | (Non-stationary case) action density as function of D, S, X, Y at time T + DT. |
| warea | The big array, used in data pool scheme, to contain many variables. |
| lwarea | *Warea* for logical variable storage. |
| compda | Array containing depth and other arrays of (*ix, iy*). |
| swtsda | Intermediate data computed for the test points. |
| xytst | Test points. |
| it | Timestep counter for SWAN. |
| kgrpnt | Grid point addresses. |
| obsta | Array of obstacle parameters. |
| cross | Array which contains 0's if there is no obstacle crossing if an obstacle is crossing between the central point and its neighbor *cross* is equal to the number of the obstacle. |

#### 5.3.6.14 Subroutine SWOMPU

Subroutine SWOMPU computes the wave spectrum for one sweep direction and is called four times per iteration.

**Calling Sequence:** swompu (swpdir, ksx, ksy, ix, iy, ddx, ddy, dt, snlc1, dal1, dal2, dal3, xis, swtsda, inocnv, ac2, compda, spcdir, spcsig, xytst, iter, warea, cgo, cg, cax, cay, cas, cad, swmatr, lswmat, kwave, alimw, groww, af11, ue, sa1, sa2, da1c, da1p, da1m, da2c, da2p, da2m, sfnl, dsnl, memnl4, idcmin, idcmax, sector, wwint, wwawg, wwswg, icolu2, diflow, difdig, difupp, difrhv, band, exact, rhv, rinsol, solut, work, precon, upperi, loperi, infmat, iinsol, iscmin, iscmax, anywnd, ac1, it, precor, xcgrid, ycgrid, kgrpnt, cross, obsta, obredf, cax1, cay1)

**Data Declaration:**

| | |
|---|---|
| Integer | iter, it, ix, iy, swpdir, ksx, ksy, inocnv, xytst, warea, idcmin, idcmax, iscmin, iscmax, sector, wwint, infmat, iinsol, kgrpnt, obsta, cross |
| Real | spcdir, spcsig, xcgrid, ycgrid, ddx, ddy, dt, dal1, |

| | | |
|---|---|---|
| | | dal2, dal3, xis, ac2, ac1, compda, cgo, cg, cax, cay, cax1, cay1, cas, cad, alimw, swmatr, kwave, af11, ue, sa1, sa2, da1c, da1p, da1m, da2c, da2p, da2m, sfnl, dsnl, memnl4, swtsda, wwawg, wwswg, icolu2, diflow, difdig, difupp, difrhv, band, exact, rhv, rinsol, solut, work, precon, upperi, loperi, obredf |
| | Logical | lswmat, groww, anywnd, precor |
| **Arguments:** | iter | Iteration counter for SWAN. |
| | it | Timestep counter for SWAN. |
| | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |
| | lswmat | Logical equivalence of *swmatr*. |
| | swpdir | Current sweep direction. |
| | ksx | Dummy variable to get the right sign in the numerical difference scheme in x-direction. |
| | ksy | Dummy variable to get the right sign in the numerical difference scheme in y-direction. |
| | ix | Counter of grid points in x-direction. |
| | iy | Counter of grid points in y-direction. |
| | ddx | Length of spatial cell in x-direction, but with correct sign depending of the direction of the sweep (+1 or -1). |
| | ddy | Length of spatial cell in y-direction, but with the correct sign depending of the direction of the sweep (+1 or -1). |
| | dt | Timestep. |
| | snlc1 | Coefficient for the subroutine SWSNLN. |
| | dal1, dal2, dal3 | Lambda dependent weight factors. |
| | xis | Difference between succeeding frequencies. |
| | swtsda | Intermediate data computed for the test points. |
| | inocnv | Counts occurrence of nonconvergence in solver. |
| | ac2 | (Non-stationary case) action density as function of D, S, X, Y at time T + DT. |
| | compda | Array containing depth and other arrays of (*ix, iy*). |

| | |
|---|---|
| xytst | Test points. |
| warea | The big array, used in data pool scheme, to contain many variables. |
| cgo | Group velocity as function of *ic* and *is* in the direction of wave propagation in absence of currents. |
| cg | Group velocity as function of *ic, is* and *id* in the direction of wave propagation in presence of currents. |
| cax | Wave transport velocity in x-direction, function of (*id, is, ic*). |
| cay | Wave transport velocity in y-direction, function of (*id, is, ic*). |
| cas | Wave transport velocity in s-direction, function of (*id, is, ic*). |
| cad | Wave transport velocity in d-direction, function of (*id, is, ic*). |
| swmatr | An array containing several variables (for data pool). |
| kwave | Wave number as function of the relative frequency S and position *ic (ix, iy)*. |
| alimw | Maximum energy by wind growth. This dummy array is used because the maximum value has to be checked directly after the solver of the tri-diagonal matrix. |
| groww | Check for a certain frequency if the waves are growing or not in a spectral direction. |
| af11 | Scaling frequency. |
| ue | "Unfolded" spectrum. |
| sa1, sa2 | Interaction contribution of first and second quadrants, respectively (unfolded space). |
| da1c, da1p, da1m, da2c, da2p, da2m | Items for diagonal matrix. |
| sfnl | Source term Snl, RHS part. |
| dsnl | Source term Snl, DIAG part. |
| memnl4 | Saves sfnl at every x,y point in memory. |
| idcmin | Frequency dependent counter in directional space. |
| idcmax | Frequency dependent counter in directional space. |
| sector | Indicates which configuration is present. |
| wwint | Counters for four wave-wave interactions. |
| wwawg | Weight coefficients for the four wave-wave interactions. |
| wwswg | Weights coefficients for the four wave-wave interactions for the semi-implicit computation. |

| | |
|---|---|
| icolu2 | Auxiliary array for storing the coefficients in the last column. |
| diflow | Lower diagonal in solver for diffusion. |
| difdig | Diagonal in solver for diffusion. |
| difupp | Upper diagonal in solver for diffusion. |
| difrhv | Right-hand vector. |
| band | Matrix from the equations to be solved. |
| exact | Exact solution. |
| rhv | Right-hand side. |
| rinsol | Real information for the solver. |
| solut | Iterative solution. |
| work | Work space. |
| precon | Preconditioner. |
| upperi | Only relevant for computation in periodic domain. |
| loperi | Only relevant for computation in periodic domain. |
| infmat | Integer information for the matrix. |
| iinsol | Integer information for the solver. |
| iscmin | Frequency dependent counter in frequency space. |
| iscmax | Frequency dependent counter in frequency space. |
| anywnd | Indicator if wind input has to be taken into account for a bin. |
| ac1 | Action density as function of D, S, X, Y at time T. |
| precor | Determines whether first guess estimate for stationary mode is calculated. |
| kgrpnt | Grid point addresses. |
| cross | Array which contains 0's if there is no obstacle crossing if an obstacle is crossing between the central point and its neighbor *cross* is equal to the number of the obstacle. |
| obsta | Array of obstacle parameters. |
| obredf | Array of action density reduction coefficients. |
| cax1 | Propagation velocity in x old time level. |
| cay1 | Propagation velocity in y old time level. |

### *5.3.7 Source Terms and Dissipation Subroutines (swancom2 FOR File)*

#### 5.3.7.1 Subroutine BRKPAR

Subroutine BRKPAR determines the bottom slope in upwave direction and calculates the slope dependent breaking parameter according to Nelson (1987). It is used here because Nelson (1994) has an error present in the equation.

**Calling Sequence:** brkpar (mdc, msc, ecos, esin, pi, ac2, spcsig, dep2, psurf, msurf, icmax, etot, kcgrd, mcgrd, rdx, rdy)

| | | |
|---|---|---|
| **Data Declaration:** | Real | ac2, ecos, esin, dep2, psurf, rdx, rdy, etot, spcsig, pi |
| | Integer | msc, mdc, msurf, kcgrd, mcgrd, icmax |

| | | |
|---|---|---|
| **Arguments:** | mdc | Maximum counter of directional distribution. |
| | msc | Maximum counter of relative frequency. |
| | ecos | Cosine of angle. |
| | esin | Sine of angle. |
| | pi | 3.14. |
| | ac2 | Action density. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | dep2 | Depth. |
| | psurf | Coefficients for breaking module. |
| | msurf | Dimensioning size for *psurf.* |
| | icmax | Maximum number of elements in *kcgrd.* |
| | etot | Total wave energy density in a particular direction. |
| | kcgrd | Grid counter in central grid point. |
| | mcgrd | Maximum counter in geographical space. |
| | rdx, rdy | Array containing spatial derivative coefficient. |

### 5.3.7.2 Subroutine FRABRE

Subroutine FRABRE computes the fraction of breaking waves in point (*ix, iy*) of the computational grid.

**Calling Sequence:** frabre (hm, etot, qbloc)

**Data Declaration:** Real hm, etot, qbloc

| | | |
|---|---|---|
| **Arguments:** | etot | Total energy per spatial grid point. |
| | qbloc | Second iteration of the fraction of breaking waves. |
| | hm | Maximum wave height. |

### 5.3.7.3 Subroutine FRABRE2

Subroutine FRABRE2 computes the fraction of breaking waves in point (*ix, iy*) of the computational grid.

**Calling Sequence:** frabre2 (hm, etot, qbloc)

**Data Declaration:** Real hm, etot, qbloc

| | | |
|---|---|---|
| **Arguments:** | etot | Total energy per spatial grid point. |
| | qbloc | Second iteration of the fraction of breaking waves. |
| | hm | Maximum wave height. |

### 5.3.7.4 Subroutine PLTSRC

Subroutine PLTSRC stores the source terms for the TESTFL grid point in a file.

**Calling Sequence:** pltsrc (plwnda, plwndb, plwcap, plbtfr, plwbrk, plnl4s, plnl4d, pltri, ac2, spcsig, dep2, xytst, kgrpnt)

| | | |
|---|---|---|
| **Data Declaration:** | Real | ac2, spcsig, plwnda, plwndb, plwcap, plbtfr, plwbrk, plnl4s, plnl4d, pltri, dep2 |
| | Integer | xytst, kgrpnt |

| | | |
|---|---|---|
| **Arguments:** | plwnda | Value of source term for test point. |
| | plwndb | Value of source term for test point. |
| | plwcap | Array containing the whitecapping source term for test output. |
| | plbtfr | For outputting on of the source terms at a particular grid point. |
| | plwbrk | For outputting on of the source terms at a particular grid point. |
| | plnl4s | For outputting on of the source terms at a particular grid point. |
| | plnl4d | For outputting on of the source terms at a particular grid point. |
| | pltri | Value of the triad source terms in test points. |
| | ac2 | Action density. |
| | spcsig | Relative frequencies in the computational domain in sigma space. |
| | dep2 | Depth. |
| | xytst | Test points. |
| | kgrpnt | Grid point addresses. |

### 5.3.7.5 Subroutine SBOT

Subroutine SBOT provides computation of the source terms due to bottom friction.

**Calling Sequence:** sbot (mdc, msc, icmax, icur, ibot, grav, abrbot, dep2, ecos, esin, imatda, kwave, spcsig, ubot, ux2, uy2, pbot, mbot, idcmin, idcmax, plbtfr, isstop, dissc1, varfr, frcoef, kcgrd, mcgrd)

| | | |
|---|---|---|
| **Data Declaration:** | Real | spcsig, grav, abrbot, dep2, ecos, esin, imatda, kwave, pbot, plbtfr, ubot, ux2, uy2, dissc1, frcoef |
| | Integer | icur, ibot, mdc, msc, icmax, mbot, isstop, mcgrd, kcgrd, idcmin, idcmax |
| | Logical | varfr |

| | | |
|---|---|---|
| **Arguments:** | spcsig | Relative frequencies in the computational domain in sigma space. |
| | mdc | Maximum counter of directional distribution. |
| | msc | Maximum counter of relative frequency. |
| | icmax | Maximum counter for the points of the molecule. |
| | icur | Indicator if a current is present. |
| | ibot | Indicator if bottom friction is on. |
| | grav | Gravitational acceleration. |
| | abrbot | Near bottom excursion amplitude. |
| | dep2 | Depth. |
| | ecos | Cosine per spectral direction (id). |
| | esin | Sine per spectral direction (id). |
| | imatda | Coefficients of diagonal of matrix. |
| | kwave | Wave number function of frequency and *ic*. |
| | ubot | Near bottom velocity as function of X, Y. |
| | ux2 | Current velocity in x direction as function of X, Y. |
| | uy2 | Current velocity in y direction as function of X, Y. |
| | pbot | Coefficient for bottom friction models. |
| | mbot | Maximum array size for the array *pbot.* |
| | idcmin | Minimum number for counter *iddum.* |
| | idcmax | Maximum number for counter *iddum.* |
| | plbtfr | For outputting on of the source terms at a particular grid point. |
| | isstop | Maximum counter of wave component in frequency space that is propagated. |
| | dissc1 | Dissipation coefficient, function of sigma and theta. |
| | varfr | Friction is spatially varying. |
| | frcoef | Spatially variable friction coefficient. |
| | kcgrd | Grid counter in central grid point. |
| | mcgrd | Maximum counter in geographical space. |

### 5.3.7.6 Subroutine SSURF

Subroutine SSURF provides computation of the source term due to wave breaking. Whitecapping is not taken into account.

**Calling Sequence:** ssurf (etot, hm, qb, smebrk, ac2, imatra, imatda, idcmin, idcmax, plwbrk, isstop, dissc0, dissc1)

| | | |
|---|---|---|
| **Data Declaration:** | Real | ac2, dissc0, dissc1, imatda, imatra, plwbrk, etot, hm, qb, smebrk |
| | Integer | isstop, idcmin, idcmax |

| | | |
|---|---|---|
| **Arguments:** | ac2 | Action density array. |
| | dissc0 | (Not used); Stores the dissipation distributed over spectral space in one point of the computational grid (old value). |
| | dissc1 | (Not used); Dissipation coefficient, function of sigma and theta. |
| | etot | Total energy per spatial grid point. |
| | hm | Maximum wave height. |
| | idcmin | Minimum number for counter *iddum.* |
| | idcmax | Maximum number for counter *iddum.* |
| | imatda | Coefficient of diagonal matrix. |
| | imatra | Coefficient of the right-hand side of the matrix. |
| | isstop | Maximum for counter *is.* |
| | plwbrk | For outputting on of the source terms at a particular grid point. |
| | qb | Fraction of breaking waves. |
| | smebrk | Mean frequency according to first order moment. |

### 5.3.7.7 Subroutine SWCAP

Subroutine SWCAP calculates the dissipation due to whitecapping.

**Calling Sequence:** swcap (spcdir, spcsig, kwave, ac2, idcmin, idcmax, isstop, etot, imatda, imatra, plwcap, dep2)

| | | |
|---|---|---|
| **Data Declaration:** | Real | ac2, dep2, etot, kwave, spcdir, spcsig, plwcap, imatda, imatra |
| | Integer | isstop, idcmin, idcmax |

| | | |
|---|---|---|
| **Arguments:** | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | kwave | Wave number. |

| | |
|---|---|
| ac2 | Action density array. |
| idcmin | Minimum number for counter *iddum*. |
| idcmax | Maximum number for counter *iddum*. |
| isstop | Maximum for counter *is*. |
| etot | Total energy per spatial grid point. |
| imatda | Coefficient of diagonal matrix. |
| imatra | Coefficient of right-hand side of matrix. |
| plwcap | Array containing the whitecapping source term for test output. |
| dep2 | Array containing water depth. |

### *5.3.8 Source Terms for Generation of Wave Energy Subroutines (swancom3 FOR File)*

#### 5.3.8.1 Subroutine SWIND0

Subroutine SWIND0 provides computation of the source term for the wind input for a third generation wind growth model: Linear wind input term according to Cavaleri and Malanotte-Rizzoli (1981).

**Calling Sequence:** swind0 (mdc, msc, idcmin, idcmax, isstop, spcsig, thetaw, grav, pi, anywnd, ufric, fpm, plwnda, imatra, spcdir, kcgrd, icmax, pwind)

**Data Declaration:**

| | |
|---|---|
| Real | fpm, grav, ufric, thetaw, pi, imatra, plwnda, pwind, spcdir, spcsig |
| Integer | mdc, msc, idcmin, idcmax, isstop, kcgrd |
| Logical | anywnd |

**Arguments:**

| | |
|---|---|
| mdc, msc | Counters in spectral space. |
| idcmin | Frequency dependent minimum counter. |
| idcmax | Frequency dependent maximum counter. |
| isstop | Maximum frequency that fall within a sweep. |
| spcsig | Relative frequencies in computational domain in sigma space. |
| thetaw | Mean direction of the relative wind vector. |
| grav | Gravitational acceleration. |
| pi | 3.14. |
| anywnd | Indicator if wind input has to be taken into account for a bin. |
| ufric | Wind friction velocity. |
| fpm | PM frequency. |
| plwnda | Values of source term for test point. |
| imatra | Coefficients of right-hand side of vector. |

| | | |
|---|---|---|
| spcdir | (*,1) | Spectral directions (radians); |
| | (*,2) | Cosine of spectral directions; |
| | (*,3) | Sine of spectral directions; |
| | (*,4) | Cosine^2 of spectral directions; |
| | (*,5) | Cosine*sine of spectral directions; |
| | (*,6) | Sine^2 of spectral directions. |
| kcgrd | | Grid counter in central grid point. |
| icmax | | Maximum counter for the points of the molecule. |
| pwind | | Coefficient for the wind growth model. |

### 5.3.8.2 Subroutine SWIND3

Subroutine SWIND3 provides computation of the source term for the wind input for a third generation wind growth model:

> Exponential input term, (Snyder et al. 1981, which expression has been modified by Komen et al. 1984). This input term should be combined with the dissipation term of Komen et al. (1984).

**Calling Sequence:** swind3 (mdc, msc, spcsig, thetaw, imatda, pwind, mwind, kwave, imatra, pi, idcmin, idcmax, ac2, icmax, ufric, fpm, plwndb, isstop, spcdir, anywnd, kcgrd, mcgrd)

**Data Declaration:**

| | |
|---|---|
| Real | spcsig, spcdir, fpm, ufric, thetaw, pi, ac2, imatda, kwave, pwind, plwndb |
| Integer | mdc, msc, icmax, mwind, isstop, mcgrd, kcgrd, idcmin, idcmax |
| Logical | anywnd |

**Arguments:**

| | |
|---|---|
| mdc, msc | Counters in spectral space. |
| spcsig | Relative frequencies in computational domain in sigma space. |
| thetaw | Mean direction of the relative wind vector. |
| imatda | Coefficients of the diagonal. |
| pwind | Coefficient for thw wind growth model. |
| mwind | Maximum array size for *pwind*. |
| kwave | Wave number. |
| imatra | Coefficients of right-hand side of matrix. |
| pi | 3.14. |
| idcmin | Frequency dependent minimum counter. |
| idcmax | Frequency dependent maximum counter. |
| ac2 | Action density as function of X, Y, S, and T. |
| icmax | Maximum counter for the points of the molecule. |
| ufric | Wind friction velocity. |

| | |
|---|---|
| fpm | PM frequency. |
| plwndb | Values of source term for test point. |
| isstop | Maximum frequency that fall within a sweep. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| anywnd | Indicator if wind input has to be taken into account for a bin. |
| kcgrd | Grid counter in central grid point. |
| mcgrd | Maximum counter in geographical space. |

### 5.3.8.3 Subroutine SWIND4

Subroutine SWIND4 provides computation of the source term for the wind input for a third generation wind growth model:

> Computation of the exponential input term based on a quasi-linear theory developed by Janssen (1989, 1991a). This formulation should be used in combination with the whitecapping dissipation source term according to Janssen (1991a and b) and Mastenbroek et al. (1993).

**Calling Sequence:** swind4 (mdc, msc, icmax, idwmin, idwmax, spcsig, wind10, thetaw, pwind, xis, mwind, dd, kwave, grav, imatra, pi, idcmin, idcmax, ac2, ufric, plwndb, isstop, iter, ustar, zelen, spcdir, anywnd, nstatc, it, precor, kcgrd, mcgrd)

**Data Declaration:**

| | |
|---|---|
| Real | spcsig, spcdir, grav, thetaw, wind10, ufric, ac2, imatra, kwave, pwind, plwndb, ustar, zelen, pi, xis, dd |
| Integer | idwmax, idwmin, mdc, msc, isstop, icmax, mwind, mcgrd, nstatc, kcgrd, idcmin, idcmax, it |
| Logical | anywnd, precor |

**Arguments:**

| | |
|---|---|
| mdc, msc | Counters in spectral space. |
| icmax | Maximum counter for the points of the molecule. |
| idwmin | Minimum counter for spectral wind direction. |
| idwmax | Maximum counter for spectral wind direction. |
| spcsig | Relative frequencies in computational domain in sigma space. |
| thetaw | Mean direction of the relative wind vector. |
| wind10 | Velocity of the relative wind vector. |

| | |
|---|---|
| pwind | Coefficient for the wind growth model. |
| xis | Difference between succeeding frequencies. |
| mwind | Maximum array size for *pwind.* |
| dd | Directional band width. |
| kwave | Wave number. |
| grav | Gravitational acceleration. |
| imatra | Coefficients of the right-hand side of matrix. |
| pi | 3.14. |
| idcmin | Frequency dependent minimum counter. |
| idcmax | Frequency dependent maximum counter. |
| ac2 | Action density as function of X, Y, S, and T. |
| ufric | Wind friction velocity. |
| plwndb | Values of source term for test point. |
| isstop | Maximum frequency that fall within a sweep. |
| iter | Iteration counter for SWAN. |
| ustar | Friction velocity at previous iteration level. |
| zelen | Roughness length at previous iteration level. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| anywnd | Indicator if wind input has to be taken into account for a bin. |
| nstatc | Indicator if computation is stationary. |
| it | Timestep counter for SWAN. |
| precor | Determines whether first guess estimate for stationary mode is calculated. |
| kcgrd | Grid counter in central grid point. |
| mcgrd | Maximum counter in geographical space. |

#### 5.3.8.4 Subroutine SWIND5

Subroutine SWIND5 provides computation of the source term for the wind input for a third generation wind growth model:

> The exponential input term is according to Yan (1987). This input term is valid for the higher frequency part of the spectrum (strongly forced wave components). The expression reduces to the Snyder (1981) expression form for spectral wave components with weak wind forcing and to the Plant (1982) form for more strongly forced wave components.

| | | |
|---|---|---|
| **Calling Sequence:** | swind5 (mdc, msc, spcsig, thetaw, isstop, ufric, kwave, imatra, pi, idcmin, idcmax, ac2, icmax, anywnd, plwndb, spcdir, kcgrd, mcgrd) | |
| **Data Declaration:** | Real | spcsig, spcdir, ac2, pi, ufric, thetaw, imatra, kwave, plwndb |
| | Integer | kcgrd, mcgrd, idcmin, idcmax, icmax, isstop, mdc, msc |
| | Logical | anywnd |
| **Arguments:** | mdc, msc | Counters in spectral space. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | thetaw | Mean direction of the relative wind vector. |
| | isstop | Maximum frequency that fall within a sweep. |
| | ufric | Wind friction velocity. |
| | kwave | Wave number. |
| | imatra | Coefficients of right-hand side of matrix. |
| | pi | 3.14. |
| | idcmin | Frequency dependent minimum counter. |
| | idcmax | Frequency dependent maximum counter. |
| | ac2 | Action density as function of X, Y, S, and T. |
| | icmax | Maximum counter for the points of the molecule. |
| | anywnd | Indicator if wind input has to be taken into account for a bin. |
| | plwndb | Values of source term for test point. |
| | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| | kcgrd | Grid counter in central grid point. |
| | mcgrd | Maximum counter in geographical space. |

### 5.3.8.5 Subroutine WNDPAR

Subroutine WNDPAR provides computation of the wind input source term with formulations of a first-generation model (constant proportionality coefficient) and a second-generation model (proportionality coefficient depends on the energy in the wind sea part of the spectrum). The expressions are from Holthuijsen and De Boer (1988) and from the DOLPHIN-B model. During the implementation of the terms, modifications to the code have been made after personal communications with Holthuijsen and Booij.

| | | |
|---|---|---|
| **Calling Sequence:** | wndpar (isstop, idwmin, idwmax, idcmin, idcmax, dep2, wind10, thetaw, ac2, kwave, imatra, imatda, spcsig, cgo, alimw, groww, etotw, plwnda, plwndb, spcdir, iter) | |
| **Data Declaration:** | Real | spcdir, spcsig, wind10, thetaw, etotw, ac2, alimw, imatda, imatra, kwave, plwnda, plwndb, dep2, cgo |
| | Integer | iter, idwmin, idwmax, iddum, isstop, idcmin, idcmax |
| | Logical | groww |
| **Arguments:** | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | isstop | Counter for the maximum frequency of all directions. |
| | idwmin | Minimum counter for spectral wind direction. |
| | idwmax | Maximum counter for spectral wind direction. |
| | idcmin | Minimum counter in directional space. |
| | idcmax | Maximum counter in directional space. |
| | dep2 | Depth. |
| | wind10 | Velocity of the relative wind vector. |
| | thetaw | Mean direction of the relative wind vector. |
| | ac2 | Action density as function of D, S, X, Y and T. |
| | kwave | Wave number. |
| | imatra | Coefficient of right-hand side of vector. |
| | imatda | Coefficient of the diagonal. |
| | cgo | Group velocity. |
| | alimw | Limiting spectrum in terms of action density. |
| | groww | Array to determine whether there are wave generation conditions. |
| | etotw | Total energy of the wind sea part of the spectrum. |
| | plwnda | Value of source term for test point. |
| | plwndb | Value of source term for test point. |
| | iter | Iteration counter for SWAN. |

### 5.3.8.6 Subroutine WINDP1

Subroutine WINDP1 provides computation of parameters derived from the wind for several subroutines such as SWIND1, SWIND2, SWIND3 and CUTOFF.

| | | |
|---|---|---|
| **Calling Sequence:** | windp1 (wind10, thetaw, idwmin, idwmax, fpm, ufric, wx2, wy2, anywnd, spcdir, ux2, uy2, spcsig | |

| | | |
|---|---|---|
| **Data Declaration:** | Real | spcsig, spcdir, wind10, thetaw, ufric, fpm, wx2, wy2, ux2, uy2 |
| | Integer | idwmin, idwmax |
| | Logical | anywnd |

| | | |
|---|---|---|
| **Arguments:** | wind10 | Velocity of the relative wind vector. |
| | thetaw | Mean direction of the relative wind vector. |
| | idwmin | Minimum counter for spectral wind direction. |
| | idwmax | Maximum counter for spectral wind direction. |
| | fpm | PM frequency. |
| | ufric | Wind friction velocity. |
| | wx2, wy2 | Wind velocity array relative to a current. |
| | anywnd | Indicator if wind input has to be taken into account for a bin. |
| | ux2 | (Non-stationary case) X-component of current velocity in (X, Y) at time T + DIT. |
| | uy2 | (Non-stationary case) Y-component of current velocity in (X, Y) at time T + DIT. |
| | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| | spcsig | Relative frequencies in computational domain in sigma space. |

### 5.3.8.7 Subroutine WINDP2

Subroutine WINDP2 provides computation of the wind sea energy spectrum for the second-generation wind growth model.

| | | |
|---|---|---|
| **Calling Sequence:** | windp2 (idwmin, idwmax, sigpkd, fpm, etotw, ac2, spcsig, wind10) | |

| | | |
|---|---|---|
| **Data Declaration:** | Integer | idwmin, idwmax |
| | Real | spcsig, etotw, fpm, ac2, sigpkd, wind10 |

| | | |
|---|---|---|
| **Arguments:** | idwmin | Minimum counter for spectral wind direction. |
| | idwmax | Maximum counter for spectral wind direction. |

| | |
|---|---|
| sigpkd | Adapted peak frequency for shallow water. |
| fpm | PM frequency. |
| etotw | Total energy of the wind sea part of the spectrum. |
| ac2 | Action density as function of D, S, X, Y and T. |
| spcsig | Relative frequencies in computational domain in sigma space. |
| wind10 | Velocity of the relative wind vector. |

### 5.3.8.8 Subroutine WINDP3

Subroutine WINDP3 reduces the energy density in the spectral direction directly after solving the tri-diagonal matrix, if the energy density level is larger than the upper bound limit given by a Pierson Moskowitz spectrum. This is only carried out if a particular wave component is "growing". If the energy density in a bin is larger than the upper bound limit (for instance when crossing wind seas are present) then the energy density level is a lower limit.

**Calling Sequence:** windp3 (mdc, msc, isstop, alimw, ac2, groww, idcmin, idcmax, kcgrd, mcgrd, icmax)

**Data Declaration:**

| | |
|---|---|
| Real | ac2, alimw |
| Integer | mdc, msc, mcgrd, icmax, idcmin, idcmax, kcgrd, isstop |
| Logical | groww |

**Arguments:**

| | |
|---|---|
| mdc, msc | Counters in spectral space. |
| isstop | Maximum frequency that falls within a sweep. |
| alimw | Contains the action density upper bound limit regarding spectral action density per spectral bin (A(s, t)). |
| ac2 | Action density as function of X, Y, S, and T. |
| groww | Logical array which determines if there is<br>a) generation ($E < E\_lim \rightarrow$ True) or<br>b) dissipation ($E > E\_lim \rightarrow$ False). |
| idcmin | Frequency dependent minimum counter. |
| idcmax | Frequency dependent maximum counter. |
| kcgrd | Grid counter in central grid point. |
| mcgrd | Maximum counter in geographical space. |
| icmax | Maximum counter for the points of the molecule. |

### *5.3.9 Nonlinear Four Wave-wave Interaction Subroutines (swancom4 FOR File)*

#### 5.3.9.1 Subroutine BND4WW

Subroutine BND4WW computes the array size for the nonlinear four-wave interactions in order to allocate some memory in the *warea.*

**Calling Sequence:** bnd4ww (mscmax, mdcmax, spcsig)

| | | |
|---|---|---|
| **Data Declaration:** | Real | spcsig |
| | Integer | mscmax, mdcmax |
| **Arguments:** | mscmax | Auxiliary variable for the 4-WAVE interactions to allocate required memory in the *warea.* |
| | mdcmax | Auxiliary variable for the 4-WAVE interactions to allocate required memory in the *warea.* |
| | spcsig | Relative frequencies in computational domain in sigma space. |

#### 5.3.9.2 Subroutine FAC4WW

Subroutine FAC4WW calculates interpolation constants for *snl.*

**Calling Sequence:** fac4ww (iter, xis, snlc1, dal1, dal2, dal3, spcsig, af11, wwint, wwawg, wwswg)

| | | |
|---|---|---|
| **Data Declaration:** | Real | spcsig, af11, xis, snlc1, wwawg, wwswg, dal1, dal2, dal3 |
| | Integer | iter, wwint |
| **Arguments:** | iter | Iteration number. |
| | xis | Difference between succeeding frequencies. |
| | snlc1 | Coefficient for the subroutines SWSNLN. |
| | dal1, dal2, dal3 | Lambda dependent weight factors. |
| | spcsig | Relative frequencies in computational domain sigma space. |
| | af11 | Scaling frequency. |
| | wwint | Counters for four-wave interactions. |
| | wwawg | Values for the interpolation. |
| | wwswg | Values for the interpolation. |

### 5.3.9.3 Subroutine FILNL3

Subroutine FILNL3 fills the *imatra* array with the nonlinear wave-wave interaction source term for a grid point (*ix, iy*) per sweep direction.

| | | |
|---|---|---|
| **Calling Sequence:** | filnl3 (mdc, msc, idcmin, idcmax, imatra, memnl4, plnl4s, isstop, kcgrd, mcgrd, icmax) | |
| **Data Declaration:** | Real | imatra, memnl4, plnl4s |
| | Integer | mdc, msc, idcmin, idcmax, isstop, kcgrd, mcgrd, icmax |
| **Arguments:** | mdc | Grid points in theta-direction of computational grid. |
| | msc | Grid points in sigma-direction of computational grid. |
| | idcmin | Minimum frequency dependent counter in directional space. |
| | idcmax | Maximum frequency dependent counter in directional space. |
| | imatra | Coefficient of the right-hand side of the matrix. |
| | memnl4 | Saves sfnl at every x,y point in memory. |
| | plnl4s | For outputting on of the source terms at a particular grid point. |
| | isstop | Maximum frequency that is propagated within a sweep. |
| | kcgrd | Grid address of points of computational stencil. |
| | mcgrd | Number of wet grid points of the computational grid. |
| | icmax | Number of points in computational stencil. |

### 5.3.9.4 Subroutine RANGE4

Subroutine RANGE4 calculates the minimum and maximum counters in frequency and directional space that fall with the calculation for the nonlinear wave-wave interactions.

| | | |
|---|---|---|
| **Calling Sequence:** | range4 (wwint, iddlow, iddtop) | |
| **Data Declaration:** | Integer | wwint, iddlow, iddtop |
| **Arguments:** | wwint | Counters for four-wave interactions. |
| | iddlow | Minimum counter of the bin that is propagated within a sweep. |
| | iddtop | Maximum counter of the bin that is propagated within a sweep. |

### 5.3.9.5 Subroutine STRIAD

Subroutine STRIAD models the triad self-interaction based on Boussinesq equation.

**Calling Sequence:** striad (ac2, dep2, cgo, imatra, kwave, hs, iddlow, iddtop, spcsig, smebrk, imatda, pltri, ursell)

**Data Declaration:**

| | | |
|---|---|---|
| Real | ac2, dep2, cgo, imatra, kwave, hs, spcsig, imatda, ursell, pltri |
| Integer | iddlow, iddtop |

**Arguments:**

| | |
|---|---|
| ac2 | Action density as function of D, S, X, Y at time T. |
| dep2 | Depth at (*ix, iy*). |
| cgo | Group velocity. |
| imatra | Right-hand vector. |
| kwave | Wave number. |
| hs | Significant wave height. |
| iddlow | Minimum counter in directional space. |
| iddtop | Maximum counter in directional space. |
| spcsig | Relative frequencies in computational domain in sigma space. |
| smebrk | Mean frequency according to first order moment. |
| imatda | Coefficient of diagonal matrix. |
| pltri | Values of the triad source terms in test points. |
| ursell | *Ursell* number as function of *ix* and *iy*. |

### 5.3.9.6 Subroutine STRIAN

Subroutine STRIAN calculates triad-wave interactions with the LTA of Eldeberky (1996). His expression that is based on a parameterization of the biphase (in terms of the *ursell* number) is directionally uncoupled and takes into account for self-self interactions only. For a full description of the equations reference is made to Eldeberky (1996). Only the main equations are given here.

**Calling Sequence:** strian (ac2, dep2, cgo, imatra, kwave, hs, iddlow, iddtop, spcsig, smebrk, imatda, pltri, ursell)

**Data Declaration:**

| | |
|---|---|
| Real | ac2, dep2, cgo, imatra, kwave, hs, spcsig, smebrk, imatda, pltri, ursell |
| Integer | iddlow, iddtop |

| | | |
|---|---|---|
| **Arguments:** | ac2 | Action density as function of D, S, X, Y at time T. |
| | dep2 | Depth at grid point (*ix, iy*). |
| | cgo | Group velocity. |
| | imatra | Right-hand vector. |
| | kwave | Wave number. |
| | hs | Significant wave height. |
| | iddlow | Minimum counter in directional space. |
| | iddtop | Maximum counter in directional space. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | smebrk | Mean frequency. |
| | imatda | Diagonal of matrix. |
| | pltri | Values of the triad source terms in test points. |
| | ursell | *Ursell* number as function of *ix* and *iy*. |

### 5.3.9.7 Subroutine SWSNL1

Subroutine SWSNL1 calculates a non-linear interaction using the discrete interaction approximation (Hasselmann and Hasselmann 1985; WAMDI group, 1988), including the diagonal term for the implicit integration. The interactions are calculated for all bins that fall within a sweep. No additional auxiliary array is required.

**Calling Sequence:** swsnl1 (wwint, wwawg, wwswg, idcmin, idcmax, af11, ue, sa1, sa2, daic, da1p, da1m, da2c, da2p, da2m, spcsig, snlc1, kmespc, fachfr, isstop, dal1, dal2, dal3, sfnl, dsnl, dep2, ac2, imatda, imatra, plnl4s, plnl4d, iddlow, iddtop)

| | | |
|---|---|---|
| **Data Declaration:** | Real | wwawg, wwswg, spcsig, af11, da1c, da1p, da1m, da2c, da2p, da2m, sa1, sa2, ue, snlc1, dal1, dal2, dal3, sfnl, dsnl, dep2, ac2, imatda, imatra, plnl4s, plnl4d, fachfr, kmespc |
| | Integer | wwint, idcmin, idcmax, iddlow, iddtop, isstop |

| | | |
|---|---|---|
| **Arguments:** | wwint | Counters for four-wave interactions. |
| | wwawg | Values for the interpolation. |
| | wwswg | Values for the interpolation. |
| | idcmin | Minimum frequency dependent counter in directional space. |
| | idcmax | Maximum frequency dependent counter in directional space. |
| | spcsig | Relative frequencies in computational domain sigma space. |
| | af11 | Scaling frequency. |
| | ue | "Unfolded" spectrum. |

| | |
|---|---|
| sa1, sa2 | (Array) Interaction contribution of first and second quadrants, respectively (unfolded space). |
| da1c, da1p, da1m, da2c, da2p, da2m | Items for diagonal matrix. |
| snlc1 | Coefficient for the subroutines SWSNLN. |
| kmespc | Mean average wave number according to the WAM formulation. |
| fachfr | Contribution of high frequency tail to wave stress. |
| isstop | Maximum frequency that is propagated within a sweep. |
| dal1, dal2, dal3 | Lambda dependent weight factors. |
| sfnl | Source term Snl, RHS part. |
| dsnl | Source term Snl, DIAG part. |
| dep2 | Depth. |
| ac2 | Action density as function of D, S, X, Y at time T. |
| imatda | Coefficient of the diagonal of the matrix. |
| imatra | Coefficient of the right-hand side of the matrix. |
| plnl4s | For outputting on of the source terms at a particular grid point. |
| plnl4d | For outputting on of the source terms at a particular grid point. |
| iddlow | Minimum counter of the bin that is propagated within a sweep. |
| iddtop | Maximum counter of the bin that is propagated within a sweep. |

### 5.3.9.8 Subroutine SWSNL2

Subroutine SWSNL2 calculates non-linear interaction using the discrete interaction approximation (Hasselmann and Hasselmann 1985; WAMDI group, 1988).

**Calling Sequence:** swsnl2 (iddlow, iddtop, wwint, wwawg, af11, ue, sa1, isstop, sa2, spcsig, snlc1, dal1, dal2, dal3, sfnl, dep2, ac2, kmespc, imatra, fachfr, plnl4s, idcmin, idcmax)

| | | |
|---|---|---|
| **Data Declaration:** | Real | wwawg, af11, ue, sa1, sa1, spcsig, snlc1, dal1, dal2, dal3, sfnl, dep2, ac2, kmespc, imatra, fachfr, plnl4s |
| | Integer | iddlow, iddtop, wwint, isstop, idcmin, idcmax |
| **Arguments:** | iddlow | Minimum counter of the bin that is propagated within a sweep. |

| | |
|---|---|
| iddtop | Maximum counter of the bin that is propagated within a sweep. |
| wwint | Counters for four-wave interactions. |
| wwawg | Values for the interpolation. |
| af11 | Scaling frequency. |
| ue | "Unfolded" spectrum. |
| sa1, sa2 | Interaction contribution of first and second quadrants, respectively (unfolded space). |
| isstop | Maximum frequency that is propagated within a sweep. |
| spcsig | Relative frequencies in computational domain sigma space. |
| snlc1 | Coefficient for the subroutines SWSNLN. |
| dal1, dal2, dal3 | Lambda dependent weight factors. |
| sfnl | Source term Snl, RHS part. |
| dep2 | Depth. |
| ac2 | Action density as function of D, S, X, Y at time T. |
| kmespc | Mean average wave number according to the WAM formulation. |
| imatra | Coefficient of right-hand side of matrix. |
| fachfr | Contribution of high frequency tail to wave stress. |
| plnl4s | For outputting on of the source terms at a particular grid point. |
| idcmin | Minimum frequency dependent counter in directional space. |
| idcmax | Maximum frequency dependent counter in directional space. |

### 5.3.9.9 Subroutine SWSNL3

Subroutine SWSNL3 calculates non-linear interaction using the discrete interaction approximation (Hasselmann and Hasselmann 1985; WAMDI group, 1988) for the full circle (option if a current is present). Using this subroutine requires an additional array with size (MXC*MYC*MDC*MSC). Although it requires more internal memory, if a current is present, it can speed up the computations significantly.

**Calling Sequence:** swsnl3 (mdc, msc, wwint, wwawg, af11, ue, sa1, sa2, spcsig, snlc1, dal1, dal2, dal3, sfnl, dep2, ac2, kmespc, memnl4, facher, pi, msc4mi, msc4ma, mdc4mi, mdc4ma, kcgrd, mcgrd, icmax

**Data Declaration:** Real wwawg, af11, ue, sa1, sa2, spcsig, snlc1, dal1, dal2, dal3, sfnl, dep2, ac2, kmespc, memnl4, facher, pi,

| | | |
|---|---|---|
| | Integer | wwint, msc4mi, msc4ma, mdc4mi, mdc4ma, kcgrd, mcgrd, icmax, mdc, msc |
| **Arguments:** | mdc | Grid points in theta-direction of computational grid. |
| | msc | Grid points in sigma-direction of computational grid. |
| | wwint | Counters for four-wave interactions. |
| | wwawg | Values for the interpolation. |
| | af11 | Scaling frequency. |
| | ue | "Unfolded" spectrum. |
| | sa1, sa2 | Interaction contribution of first and second quadrants, respectively (unfolded space). |
| | spcsig | Relative frequencies in computational domain sigma space. |
| | snlc1 | Coefficient for the subroutine SWSNLN. |
| | dal1, dal2, dal3 | Lambda dependent weight factors. |
| | sfnl | Source term Snl, RHS part. |
| | dep2 | Depth. |
| | ac2 | Action density as function of D, S, X, Y at time T. |
| | kmespc | Mean average wave number according to the WAM formulation. |
| | memnl4 | Saves sfnl at every x,y point in memory. |
| | fachfr | Contribution of high frequency tail to wave stress. |
| | pi | 3.14. |
| | msc4mi | Lowest array counter in frequency space. |
| | msc4ma | Highest array counter in frequency space. |
| | mdc4mi | Lowest array counter in directional space. |
| | mdc4ma | Highest array counter in directional space. |
| | kcgrd | Grid address of points of computational stencil. |
| | mcgrd | Number of wet grid points of the computational grid. |
| | icmax | Number of points in computational stencil. |

### *5.3.10 Subroutines for the Propagation in X, Y, S, D Space and Parameters (swancom5 FOR File)*

#### 5.3.10.1 Subroutine ADDDIS

Subroutine ADDDIS adds dissipation and leak.

**Calling Sequence:** adddis (msc, mdc, ddir, frintf, dissxy, leakxy, ac2, anybin, disc0, disc1, leakc1, spcsig, kcgrd, mcgrd, icmax)

| | | |
|---|---|---|
| **Data Declaration:** | Real | ddir, frintf, dissxy, leakxy, ac2, disc0, disc1, leakc1, spcsig |
| | Integer | msc, mdc, mcgrd, kcgrd, icmax |
| | Logical | anybin |
| **Arguments:** | msc | Maximum counter of relative frequency. |
| | mdc | Maximum counter of directional distribution. |
| | ddir | Spectral direction band width. |
| | frintf | Frequency integration factor. |
| | dissxy | Dissipation integrated over the spectrum for each point in the computational grid. |
| | leakxy | Leak integrated over the spectrum for each point in the computation grid. |
| | ac2 | Action density as function of D, S, X, Y and T. |
| | anybin | Determines if a bin falls within a sweep. |
| | dissc0 | (Not used); Stores the dissipation distributed over spectral space in one point of the computational grid (old value). |
| | dissc1 | (Not used); Stores the dissipation distributed over spectral space in one point of the computational grid (new value). |
| | leakc1 | Leak coefficient. |
| | spcsig | Relative frequencies in the computational domain in sigma space. |
| | kcgrd | Grid counter in central grid point. |
| | mcgrd | Maximum counter in geographical space. |
| | icmax | Maximum array size for the points of the molecule. |

### 5.3.10.2 Subroutine DSPHER

Subroutine DSPHER computes the propagation velocities of energy in theta-space, i.e., CAD, due to the use of spherical coordinates.

**Calling Sequence:** dspher (cad, cg, anybin, ycgrid, ecos)

| | | |
|---|---|---|
| **Data Declaration:** | Real | cad, cg, ecos, ycgrid |
| | Logical | anybin |
| **Arguments:** | cad | Wave transport velocity in D-direction, function (*id, is, ic*). |
| | cg | Group velocity as function of sigma and theta in the direction of wave propagation in absence of currents. |

| | |
|---|---|
| anybin | If true the spectral component (*id, is*) is to be computed. |
| ycgrid | Y-coordinate (latitude) for each geographic grid point. |
| ecos | Represent the values of cos(theta) of each spectral direction. |

#### 5.3.10.3 Subroutine SANDL

Subroutine SANDL computes the space derivative of action transport.

**Calling Sequence:** sandl (isstop, idcmin, idcmax, cgo, cax, cay, ac2, ac1, imatra, imatda, rdx, rdy, cax1, cay1, spcdir)

**Data Declaration:**

| | |
|---|---|
| Real | cgo, cax, cay, ac2, ac1, imatra, imatda, rdx, rdy, cax1, cay2, spcdir |
| Integer | isstop, idcmin, idcmax |

**Arguments:**

| | |
|---|---|
| isstop | Highest spectral frequency counter in the sweep. |
| idcmin | Minimum value of direction counter in this sweep. |
| idcmax | Maximum value of direction counter in this sweep. |
| cgo | Group velocity. |
| cax | Propagation velocity in x new time level. |
| cay | Propagation velocity in y new time level. |
| ac2 | Spectral action density, function of x, y, theta, and sigma. |
| ac1 | Action density as function of D, S, X, Y at time T. |
| imatra | Coefficients of right-hand side of matrix. |
| imatda | Coefficients of diagonal of matrix. |
| rdx, rdy | Containing spatial derivative coefficient. |
| cax1 | Propagation velocity in x old time level. |
| cay1 | Propagation velocity in y old time level. |
| spcdir | Spectral directions. |

#### 5.3.10.5 Subroutine SORDUP

Subroutine SORDUP computes the space derivative of action transport using the SORDUP scheme. This is for stationary calculations only (no time derivative). Delft Hydraulics scientists suggest that the implementation of a modified form of the scheme, in which the model user has the option for using a non-zero value for THETAK, be used as a means to eliminate wiggles.

To summarize:

With THETAK = 0, the scheme is second order accurate.

With THETAK = 0, the scheme reduces to the "best" approximation of d/dx which can be determined using Taylor Series for the stencil (IX), (IX-1), (IX-2): 3/2*mu*phi(IX)-2*mu*phi(IX-1) + 1/2*mu*phi(IX-2).

With a non-zero THETAK, the scheme is only first order accurate, and is only approximately mass conserving (mass balance error is slight).

With a negative THETAK, the scheme has positive diffusion. This makes the scheme something of a hybrid between the BSBT scheme (of the original SWAN) and the second order scheme (THETAK = 0). The only reason to intentionally introduce diffusion is in case of wiggles. Wiggles will, for the most part, only occur when spatial gradients are very severe, so using a negative THETAK is generally not necessary. Using a THETAK of -0.1 for case-set of severe gradient, diffusion seems to be about midway between that of the BSBT scheme and that of the second order (THETAK = 0) scheme. For this case-set, wiggles are seen in the second order scheme solution, and are virtually eliminated with the (THETAK = -0.1) scheme. Henri has shown that the scheme with small negative THETAK is very likely to be unconditionally stable. Larger |THETAK| ==> more diffusion.

With a positive THETAK, the scheme is unconditionally unstable. This instability is generally not noticeable, but since there is not a good reason for using positive THETAK, if this option is chosen, a warning or error message will be given.

**Calling Sequence:** sordup (isstop, idcmin, idcmax, cax, cay, ac2, imatra, imatda, rdx, rdy)

**Data Declaration:**

| | | |
|---|---|---|
| Real | | cax, cay, ac2, imatra, imatda, rdx, rdy |
| Integer | | isstop, idcmin, idcmax |

**Arguments:**

| | |
|---|---|
| isstop | Highest spectral frequency counter in the sweep. |
| idcmin | Minimum value of direction counter in this sweep. |
| idcmax | Maximum value of direction counter in this sweep. |
| cax | Propagation velocity in x. |
| cay | Propagation velocity in y. |
| ac2 | Spectral action density, function of x, y, theta, sigma. |
| imatra | Coefficients of right-hand side of matrix. |
| imatda | Coefficients of diagonal of matrix. |
| rdx, rdy | Containing spatial derivative coefficient. |

#### 5.3.10.6 Subroutine SPREDT

Subroutine SPREDT predicts the action density depending on the sweep direction. A good prediction is necessary for a first accurate prediction of the action density to compute the dissipation of energy. To compute the energy dissipation a prediction is needed at time T.

| | | |
|---|---|---|
| **Calling Sequence:** | spredt (swpdir, ac2, cax, cay, idcmin, idcmax, isstop, anybin, rdx, rdy, obredf) | |
| **Data Declaration:** | Real | swpdir, ac2, cax, cay, rdx, rdy, obredf |
| | Integer | idcmin, idcmax, isstop |
| | Logical | anybin |
| **Arguments:** | swpdir | Sweep direction (identical as the description of the direction the wind is blowing). |
| | ac2 | Action density as function of D, S, X, Y at time T. |
| | cax | Wave transport velocity in x-direction, function of (*id, is, ic*). |
| | cay | Wave transport velocity in y-direction, function of (*id, is, ic*). |
| | idcmin | Minimum frequency dependent counter in case of a current. |
| | idcmax | Maximum frequency dependent counter in case of a current. |
| | isstop | Maximum frequency counter for wave components that are propagated within a sweep. |
| | anybin | Determines if a bin falls within a sweep. |
| | rdx, rdy | Array containing spatial derivative coefficient. |
| | obredf | Action reduction factors, a function of frequency and direction. |

### 5.3.10.7 Subroutine SPROSD

Subroutine SPROSD computes the propagation velocities of energy in S- and D-space, i.e., CAS, CAD, in the presence or absence of currents, for the action balance equation.

| | | |
|---|---|---|
| **Calling Sequence:** | sprosd (spcsig, kwave, cas, cad, cgo, dep2, dep1, ecos, esin, ux2, uy2, swpdir, idcmin, idcmax, coscos, sinsin, sincos, rdx, rdy, cax, cay, anybin, kgrpnt, xcgrid, ycgrid) | |
| **Data Declaration:** | Real | spcsig, kwave, cas, cad, cgo, dep2, dep1, ecos, esin, ux2, uy2, swpdir, rdx, rdy, cax, cay, xcgrid, ycgrid |
| | Integer | idcmin, idcmax, kgrpnt |
| | Logical | anybin |
| **Arguments:** | spcsig | Relative frequencies in the computational domain in sigma space. |

| | |
|---|---|
| kwave | Wave number as function of the relative frequency sigma. |
| cas | Wave transport velocity in S-direction, a function of (*id, is, ic*). |
| cad | Wave transport velocity in D-direction, a function of (*id, is, ic*). |
| cgo | Group velocity as function of X, Y and sigma in the direction of wave propagation in absence of currents. |
| dep2 | Depth as function of (X, Y) at time T+1. |
| ux2 | (Non-stationary case) X-component of current velocity in (X, Y) at time T + DIT. |
| uy2 | (Non-stationary case) Y-component of current velocity in (X, Y) at time T + DIT. |
| dep1 | Depth as function of X and Y at time T. |
| ecos | Represent the values of cos(d) of each spectral direction. |
| esin | Represent the values of sin(d) of each spectral direction. |
| swpdir | Current sweep direction. |
| idcmin | Lower theta boundary of current sweep. |
| idcmax | Upper theta boundary of current sweep. |
| coscos | Cosine^2 of spectral directions. |
| sinsin | Sine^2 of spectral directions. |
| sincos | Cosine*sine of spectral directions. |
| rdx, rdy | Array containing spatial derivative coefficient. |
| cax | Wave transport velocity in X-direction, a function of (*id, is, ic*). |
| cay | Wave transport velocity in Y-direction, a function of (*id, is, ic*). |
| anybin | = True if a certain bin is enclosed in a sweep. |
| kgrpnt | Grid point addresses. |
| xcgrid | X-coordinate of computational grid in x-direction. |
| ycgrid | Y-coordinate of computational grid in y-direction. |

### 5.3.10.8 Subroutine SPROXY

Subroutine SPROXY computes the propagation velocities of energy in X-, Y-space, i.e., *cax*, *cay*, in the presence or absence of currents, for the action balance equation. The propagation velocities are computed for the full 360 degree sector.

**Calling Sequence:** sproxy (ic, icmax, msc, mdc, icur, cax, cay, cgo, ecos, esin, ux2, uy2, swpdir, kcgrd, mcgrd)

| | | |
|---|---|---|
| **Data Declaration:** | Real | cax, cay, cgo, ecos, esin, ux2, uy2, swpdir |
| | Integer | msc, mdc, icmax, ic, icur, kcgrd, mcgrd |
| **Arguments:** | ic | Dummy variable. |
| | icmax | Maximum array size for the points of the molecule. |
| | msc | Maximum counter of relative frequency. |
| | mdc | Maximum counter of spectral directions. |
| | icur | Indicator for current. |
| | cax | Wave transport velocity in x-direction, function of (*id, is, ic*). |
| | cay | Wave transport velocity in y-direction, function of (*id, is, ic*). |
| | cgo | Group velocity. |
| | ecos | Represent the values of cos(d) of each spectral direction. |
| | esin | Represent the values of sin(d) of each spectral direction. |
| | ux2 | X-component of current velocity of X and Y at time T + 1. |
| | uy2 | Y-component of current velocity of X and Y at time T + 1. |
| | swpdir | Current sweep direction. |
| | kcgrd | Grid counter in central grid point. |
| | mcgrd | Maximum counter in geographical space. |

### 5.3.10.9 Subroutine STRSD

Subroutine STRSD computes $\partial[CAD\ AC2]/\partial D$ for the initial and boundary conditions.

**Calling Sequence:** strsd (msc, mdc, icmax, dd, idcmin, idcmax, cad, imatla, imatda, imatua, imatra, ac2, pnums, isstop, fulcir, anybin, leakc1, kcgrd, mcgrd)

| | | |
|---|---|---|
| **Data Declaration:** | Real | dd, cad, ac2, pnums, imatla, imatda, imatua, imatra, leakc1 |
| | Integer | msc, mdc, icmax, idcmin, idcmax, isstop, kcgrd, mcgrd |
| | Logical | anybin, fulcir |
| **Arguments:** | msc | Maximum counter of relative frequency. |
| | mdc | Maximum counter of directional distribution. |
| | icmax | Maximum counter for the points of the molecule. |
| | dd | Width of spectral direction band. |
| | idcmin | Minimum value of direction counter in this sweep. |

| | |
|---|---|
| idcmax | Maximum value of direction counter in this sweep. |
| cad | Wave transport velocity in S-direction, function of (*id, is, ic*). |
| imatla | Coefficients of lower diagonal of matrix. |
| imatda | Coefficients of diagonal of matrix. |
| imatua | Coefficients of upper diagonal of matrix. |
| imatra | Coefficients of right-hand side of matrix. |
| ac2 | Action density as function of D, S, X, Y at time T. |
| pnums | Array containing various coefficients/controls for the model. |
| isstop | Maximum frequency counter for wave components that are propagated within a sweep. |
| fulcir | If true, computation on a full circle. |
| anybin | = True if a certain bin is enclosed in a sweep. |
| leakc1 | Leak coefficient. |
| kcgrd | Grid counter in central grid point. |
| mcgrd | Maximum counter in geographical space. |

### 5.3.10.10 Subroutine STRSSB

Subroutine STRSSB computes ∂[CAS AC2]/∂S for the initial and boundary conditions with an explicit scheme. The energy near the blocking point is removed from the spectrum based on a CFL criterion.

**Calling Sequence:** strssb (mdc, msc, icmax, iddlow, iddtop, idcmin, idcmax, isstop, cax, cay, cas, ac2, spcsig, imatra, pnums, anyblk, kcgrd, mcgrd, rdx, rdy)

**Data Declaration:**

| | |
|---|---|
| Real | cax, cay, cas, ac2, spcsig, imatra, pnums, rdx, rdy |
| Integer | mdc, msc, icmax, iddlow, iddtop, idcmin, idcmax, isstop, kcgrd, mcgrd |
| Logical | anyblk |

**Arguments:**

| | |
|---|---|
| msc | Maximum counter of relative frequency. |
| mdc | Maximum counter of directional distribution. |
| icmax | Maximum counter for the points of the molecule. |
| iddlow | Minimum direction that is propagated within a sweep. |
| iddtop | Idem maximum. |
| idcmin | Minimum value of direction counter in this sweep. |
| idcmax | Maximum value of direction counter in this sweep. |
| isstop | Maximum frequency counter for wave components that are propagated within a sweep. |
| cax, cay | Propagation velocities in x-y space. |

| | |
|---|---|
| cas | Wave transport velocity in S-direction, function of (*id, is, ic*). |
| ac2 | Action density as function of D, S, X, Y at time T. |
| spcsig | Relative frequencies in computational domain in sigma space *imatra.* |
| pnums | Array containing various coefficients/controls for the model. |
| anyblk | Determines if a counter current blocks a bin based on a CFL criterion. |
| kcgrd | Grid counter in central grid point. |
| mcgrd | Maximum counter in geographical space. |
| rdx, rdy | Array containing spatial derivative coefficient. |

### 5.3.10.11 Subroutine STRSSI

Subroutine STRSSI computes ∂[CAS AC2]/∂S for the initial and boundary conditions with an implicit scheme.

**Calling Sequence:** strssi (msc, mdc, icmax, pnums, spcsig, cas, imat5l, imatda, imat6u, anybin, imatra, ac2, iscmin, iscmax, iddlow, iddtop, kcgrd, mcgrd)

**Data Declaration:**

| | |
|---|---|
| Real | pnums, spcsig, cas, ac2, imat5l, imatda, imat6u, imatra |
| Logical | anybin |
| Integer | msc, mdc, icmax, iscmin, iscmax, iddlow, iddtop, kcgrd, mcgrd |

**Arguments:**

| | |
|---|---|
| msc | Maximum counter of relative frequency. |
| mdc | Maximum counter of directional distribution one sweep. |
| icmax | Maximum counter for the points of the molecule. |
| pnums | Array containing various coefficients/controls for the model. |
| spcsig | Relative frequencies in computational domain in sigma space. |
| cas | Wave transport velocity in S-direction, function of (*id, is, ic*). |
| imat5l | Coefficients of lower diagonal of matrix. |
| imatda | Coefficients of diagonal of matrix. |
| imat6u | Coefficients of upper diagonal of matrix. |
| anybin | If true the spectral component (*id, is*) is to be computed. |
| imatra | Coefficients of right-hand side of matrix. |

| | |
|---|---|
| ac2 | Spectral action density, function of x, y, theta, and sigma. |
| iscmin | Minimum counter in frequency space per direction. |
| iscmax | Maximum counter in frequency space per direction. |
| iddlow | Minimum counter per sweep taken over all frequencies. |
| iddtop | Maximum counter per sweep taken over all frequencies. |
| kcgrd | Grid counter in central grid point. |
| mcgrd | Maximum counter in geographical space. |

#### 5.3.10.13 Subroutine STRSXY

Subroutine STRSXY computes the space derivative of action transport.

**Calling Sequence:** strsxy (isstop, idcmin, idcmax, cax, cay, ac2, ac1, imatra, imatda, rdx, rdy, obredf)

**Data Declaration:**

| | |
|---|---|
| Real | cax, cay, ac2, ac1, rdx, rdy, imatra, imatda, obredf |
| Integer | isstop, idcmin, idcmax |

**Arguments:**

| | |
|---|---|
| isstop | Highest spectral frequency counter in the sweep. |
| idcmin | Minimum value of direction counter in this sweep. |
| idcmax | Maximum value of direction counter in this sweep. |
| cax | Propagation velocity in x. |
| cay | Propagation velocity in y. |
| ac2 | Spectral action density, function of x, y, theta and sigma. |
| ac1 | Action density as function of D, S, X, Y at time T. |
| imatra | Coefficients of diagonal of matrix. |
| imatda | Coefficients of right-hand side of matrix. |
| rdx, rdy | Array containing spatial derivative coefficient. |
| obredf | Action reduction factors, function of frequency and direction. |

#### 5.3.10.14 Subroutine SWAPAR

Subroutine SWAPAR computes the wave parameters *k, cgo,* and *cg* in the nearby points, depending on the sweep direction. The nearby points are indicated with the index *ic*.

**Calling Sequence:** swapar (ic, msc, mdc, icmax, cg, icur, grav, dep2, kwave, cgo, ecos, esin, ux2, uy2, spcsig, kcgrd, mcgrd, depmin)

| | | |
|---|---|---|
| **Data Declaration:** | Real | cg, grav, dep2, kwave, cgo, ecos, esin, ux2, uy2, spcsig, depmin |
| | Integer | ic, msc, mdc, icmax, icur, kcgrd, mcgrd |

| | | |
|---|---|---|
| **Arguments:** | ic | Dummy variable. |
| | msc | Maximum counter of relative frequency. |
| | mdc | Maximum counter of directional distribution. |
| | icmax | Maximum array size for the points of the molecule. |
| | cg | Group velocity as function of X, Y and S and D in the direction of wave propagation in presence of currents. |
| | icur | Indicator for current. |
| | grav | Gravitational acceleration. |
| | dep2 | Depth as function of X and Y at time T+1. |
| | kwave | Wave number as a function of the relative frequency S. |
| | cgo | Group velocity as function of X, Y and S in the direction of wave propagation in the absence of currents. |
| | ecos | Represent the values of cos(d) of each spectral direction. |
| | esin | Represent the values of sin(d) of each spectral direction. |
| | ux2 | X-component of current velocity of X and Y at time T+1. |
| | uy2 | Y-component of current velocity of X and Y at time T+1. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | kcgrd | Grid counter in central grid point. |
| | mcgrd | Maximum counter in geographical space. |
| | depmin | Threshold depth (m); in the computation any positive depth smaller than *depmin* is made equal to *depmin*. Default = 0.05. |

### 5.3.10.15 Subroutine SWPSEL

Subroutine SWPSEL computes the frequency dependent counters in situations with and without a current. The counters are only computed for the grid point considered. This means *ic* = 1 (see loop with call for ICCODE function).

**Calling Sequence:** swpsel (swpdir, idcmin, idcmax, sector, cax, cay, anybin, iscmin, iscmax, idtot, istot, iddlow, iddtop, isstop, dep2, ux2, uy2, spcdir, xcgrid, ycgrid, rdx, rdy, ksx, ksy)

| | | |
|---|---|---|
| **Data Declaration:** | Real | swpdir, spcdir, xcgrid, ycgrid, sector, cax, cay, dep2, ux2, uy2, rdx, rdy, ksx, ksy |
| | Integer | idcmin, idcmax, iscmin, iscmax, idtot, istot, iddlow, iddtop, isstop |
| | Logical | anybin |

| | | |
|---|---|---|
| **Arguments:** | swpdir | Current sweep direction. |
| | idcmin | Minimum frequency dependent counter. |
| | idcmax | Maximum frequency dependent counter. |
| | sector | Counter for number enclosed sectors. |
| | cax, cay | Propagation velocities. |
| | anybin | = True if a certain bin enclosed in a sweep. |
| | iscmin | Minimum counter in frequency space. |
| | iscmax | Maximum counter in frequency space. |
| | idtot | Maximum value between the lowest and highest counter in directional space. |
| | istot | Maximum value between the lowest and highest counter in frequency space. |
| | iddlow | Minimum counter per sweep taken over all frequencies. |
| | iddtop | Maximum counter per sweep taken over all frequencies. |
| | isstop | Maximum frequency counter for wave components that are propagated within a sweep. |
| | dep2 | Depth. |
| | ux2 | (Non-stationary case) X-component of current velocity in (X, Y) at time T + DIT. |
| | uy2 | (Non-stationary case) Y-component of current velocity in (X, Y) at time T + DIT. |
| | spcdir | Spectral directions. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |
| | rdx, rdy | Array containing spatial derivative coefficient. |
| | ksx | Dummy variable to get the correct sign in the numerical difference scheme in X-direction. |
| | ksy | Dummy variable to get the right sign in the numerical difference scheme in Y-direction. |

### *5.3.11 Subroutines for Solving the Band Matrix (swancomi FOR File)*

#### 5.3.11.1 Subroutine CGSTAB

Subroutine CGSTAB solves an asymmetric system of linear equations by the Bi-CGSTAB method. The subroutine contains a number of preconditioners.

**Calling Sequence:** cgstab (n, amat, rhsd, usol, eps1, eps2, itmax, res, p, rbar, t, s, v, work, icontr, infmat, prec, nprec, ndim, nconct, upperi, loperi, nstatc, itsw, itersw)

**Data Declaration:**

| | |
|---|---|
| Real | amat, rhsd, usol, eps1, eps2, res, p, rbar, t, s, v, prec, work, upperi, loperi |
| Integer | n, itmax, icontr, infmat, nprec, ndim, nconct, nstatc, itsw, itersw |

**Arguments:**

| | |
|---|---|
| n | The number of rows in the matrix A. |
| amat | Matrix from the equations to be solved. |
| rhsd | Vector containing the right-hand side vector of the system of equations. |
| usol | Solution vector of length n. On input the array contains a starting vector. At output the array contains the last iterate, which is an approximation to the solution of the system. |
| eps1, eps2 | Determines the accuracy of the final approximation. |
| itmax | The maximum number of iterations to be performed. |
| res | Array containing the residual vector. |
| p | Work array to store the direction vector. |
| rbar | Work array to store the quasi-residual vector. |
| t, s | Work array to store an auxiliary vector. |
| v | Work array to store an auxiliary vector. |
| work | Work array to store an auxiliary vector. The array work(.,3) contains the update of the solution usol during an iteration. If post-conditioning is used, it is first adapted before it is added to *usol.* |
| icontr | Integer array in which information about the solution process must be given by the user. |
| infmat | Integer array with information of the matrix structure, to be used in matrix-vector multiplication subroutine. |
| prec | Array which contains part of the preconditioning matrix. |

| | |
|---|---|
| nprec | Number of diagonals which are used in the pre-conditioning. |
| ndim | Integer indicating the amount of unknowns in every grid point. In the momentum equations *ndim* = 2 or 3, whereas in the pressure and transport equations *ndim* = 1. |
| nconct | Maximal number of connections in one row of the matrix. |
| upperi | Only relevant for computation in periodic domain. |
| loperi | Only relevant for computation in periodic domain. |
| nstatc | Indicates stationary:<br>= 0; stationary computation;<br>= 1; non-stationary computation. |
| itsw | Timestep counter for SWAN |
| itersw | Iteration counter for SWAN. |

### 5.3.11.2 Subroutine DAXPY

Subroutine DAXPY is a BLAS routine that overwrites double precision *dy* with double precision *da*dx + dy*. For i = 0 to n-1, replace *dy(ly + i*incy)* with *da*dx(lx + i*incx) + dy(ly + i*incy)*, where lx = 1 if *incx* >= 0, else lx = *(-incx)*n*, and *ly* is defined in a similar way using *incy*.

### 5.3.11.3 Subroutine DCOPY

Subroutine DCOPY is a BLAS routine that copies double precision *dx* to double precision *dy*. For i = 0 to n-1, copy *dx(lx + i*incx)* to *dy(ly + i*incy)*, where lx = 1 if *incx* > 0, else lx = *(-incx)*n*, and ly is defined in a similar way using *incy*.

### 5.3.11.4 Double Precision Function DDOT

Subroutine DDOT calculates the dot product of two vectors of equal length.

**Calling Sequence:** ddot (dx, dy, n)

**Data Declaration:**

| | |
|---|---|
| Real | dx, dy |
| Integer | n |

| | | |
|---|---|---|
| **Arguments:** | dx | First vector in dot product. |
| | dy | Second vector in dot product. |
| | n | Vector length. |

### 5.3.11.5 Subroutine DIAG

Subroutine DIAG makes a diagonal scaling of the matrix for the momentum, transport, or pressure equations.

**Calling Sequence:** diag (amat, n, ndimso, nconct, prec, nprec, infmat)

| | | |
|---|---|---|
| **Data Declaration:** | Real | amat, prec |
| | Integer | n, ndimso, nconct, nprec, infmat |

| | | |
|---|---|---|
| **Arguments:** | amat | The coefficient matrix for the momentum equations or an equation similar to the pressure equation. |
| | n | Number of unknowns in the solution vector. |
| | ndimso | Integer indicating the dimension of the space in which the problem must be solved (*ndimso* = 1 or *ndim*). |
| | nconct | Number of connections in one row of the matrix. |
| | prec | The preconditioning matrix. |
| | nprec | Number of diagonals, which are used in the preconditioning. In this subroutine *nprec* = 1. |
| | infmat | If *infmat* = 1 momentum equations are used, whereas if *infmat* >= 4 equations with a structure similar to the pressure equation are used. |

### 5.3.11.6 Subroutine DIAGMU

Subroutine DIAGMU multiplies x with the diagonal matrix given in *prec*. The array *prec* should be filled by subroutine DIAGF.

**Calling Sequence:** diagmu (n, x, b, prec, nprec)

| | | |
|---|---|---|
| **Data Declaration:** | Real | x, b, prec |
| | Integer | n, nprec |

| | | |
|---|---|---|
| **Arguments:** | n | Number of unknowns in the solution vector. |
| | x | The original vector. |
| | b | The resulting vector after multiplication. |

| | |
|---|---|
| prec | The diagonal preconditioning matrix. |
| nprec | Number of diagonals, which are used in the preconditioning. In this subroutine *nprec* = 1. |

#### 5.3.11.7 Subroutine DINVL3

Subroutine DINVL3 multiplies x by L, the preconditioning matrix given in *prec*. In this case we obtain:

$$b = L^{-1} x.$$

The array *prec* should be filled by dmlu3.f. This subroutine contains compiler directives to run in vector speed on the convex.

**Calling Sequence:** dinvl3 (x, b, matrix, n, ndim, nconct, prec, nprec, infmat)

**Data Declaration:**

| | |
|---|---|
| Real | x, b, prec |
| Integer | matrix, n, ndim, nconct, nprec, infmat |

**Arguments:**

| | |
|---|---|
| x | The original vector. |
| b | The result vector, which contains: $b = L^{-1} x$. |
| matrix | The coefficient matrix for the momentum or an equation similar to the pressure equation. |
| n | Number of unknowns in the solution vector. |
| ndim | Integer indicating the dimension of the space in which the problem must be solved (*ndim* = 2 or 3). |
| nconct | Number of connections in one row of the matrix. |
| prec | The preconditioning matrix. |
| nprec | Number of diagonals, which are used in the preconditioning. In this subroutine *nprec* = *nconct*. |
| infmat | If *infmat* = 1 the momentum equations are used, whereas if *infmat* = 1 is larger than or equal to four equations with a structure similar to the pressure equations are used. |

#### 5.3.11.8 Subroutine DINVU3

Subroutine DINVU3 multiplies x by U, the preconditioning matrix given in *prec*. In this case we obtain:

$$b = U^{-1} x.$$

The array *prec* should be filled by dmlu3.f. This subroutine contains compiler directives to run in vector speed on the convex.

**Calling Sequence:** dinvu3 (x, b, matrix, n, ndimso, nconct, prec, nprec, infmat)

**Data Declaration:**

| | | |
|---|---|---|
| | Real | x, b, prec, matrix |
| | Integer | n, ndimso, nconct, nprec, infmat |

**Arguments:**

| | | |
|---|---|---|
| | x | The original vector. |
| | b | The result vector, which contains: $b = U^{-1} x$. |
| | matrix | The coefficient matrix for the momentum or an equation similar to the pressure equation. |
| | n | Number of unknowns in the solution vector. |
| | ndimso | Integer indicating the dimension of the space in which the problem must be solved (*ndimso* = 1 or *ndim*). |
| | nconct | Number of connections in one row of the matrix. |
| | prec | The preconditioning matrix. |
| | nprec | Number of diagonals, which are used in the preconditioning. In this subroutine *nprec* = *nconct*. |
| | infmat | If *infmat*(1) is one the momentum equations are used, whereas if *infmat*(1) is larger than or equal to four equations with a structure similar to the pressure equation are used. |

#### 5.3.11.9 Subroutine DMLU3

Subroutine DMLU3 calculates an upper triangular matrix U and a lower triangular matrix L, which form an incomplete decomposition of A.

**Calling Sequence:** dmlu3 (matrix, n, ndim, nconct, prec, nprec, infmat)

**Data Declaration:**

| | | |
|---|---|---|
| | Real | matrix, prec |
| | Integer | n, ndim, nconct, nprec, infmat |

**Arguments:**

| | | |
|---|---|---|
| | matrix | The coefficient matrix for the momentum equations or an equation similar to the pressure equation. |
| | n | Number of unknowns in the solution vector. |
| | ndim | Integer indicating the dimension of the space in which the problem must be solved (*ndim* = 2 or 3). |
| | nconct | Number of connections in one row of the matrix. |
| | prec | The preconditioning matrix. |
| | nprec | Number of diagonals, which are used in the preconditioning. In this subroutine *nprec* = *nconct*. |

| | |
|---|---|
| infmat | If *infmat*(1) is one the momentum equations are used, whereas if *infmat*(1) is larger than or equal to four, equations with a structure similar to the pressure equation are used. *Infmat*(2) is the number of discretization points in the x-direction. |

#### 5.3.11.10 Double Precision Function DNRM2

Subroutine DNRM2 calculates the Euclidean norm of a vector *dx*() of length *n*.

**Calling Sequence:** dnrm2 (n, dx, incx)

| **Data Declaration:** | | |
|---|---|---|
| | Real | dx |
| | Integer | n, incx |

| **Arguments:** | | |
|---|---|---|
| | n | Length of the vector in *dx*() |
| | dx | Array containing the vector. |
| | incx | Stride of the vector stored in *dx*(). |

#### 5.3.11.11 Subroutine DRUMA1

**Calling Sequence:** druma1 (x, b, matrix, n, nconct, infmat, upperi, loperi)

| **Data Declaration:** | | |
|---|---|---|
| | Real | x, b, matrix, upperi, loperi |
| | Integer | n, nconct, infmat |

| **Arguments:** | | |
|---|---|---|
| | x | The original vector. |
| | b | The result vector, which contains: $b = U^{-1} x$. |
| | matrix | The coefficient matrix for the momentum or an equation similar to the pressure equation. |
| | n | Number of unknowns in the solution vector. |
| | nconct | Number of connections in one row of the matrix. |
| | infmat | If *infmat*(1) is one the momentum equations are used, whereas if *infmat*(1) is larger than or equal to four, equations with a structure similar to the pressure equation are used. *Infmat*(2) is the number of discretization points in the x-direction. |
| | upperi | Only relevant for computation in periodic domain. |
| | loperi | Only relevant for computation in periodic domain. |

### 5.3.11.12 Subroutine ISSOLV

Subroutine ISSOLV solves an asymmetric system of equations of the shape Ax = f.

**Calling Sequence:** issolv (iinsol, rinsol, matrix, rhside, solut, nusol, nconct, infmat, work, nwork, precon, nprec, upperi, loperi, inocnv, itsw, itersw)

**Data Declaration:**

| | |
|---|---|
| Real | rinsol, matrix, work, rhside, solut, precon, upperi, loperi |
| Integer | iinsol, nusol, infmat, itsw, itersw, inocnv, nonct, nwork, nprec |

**Arguments:**

| | |
|---|---|
| iinsol | Integer information for the solver. |
| rinsol | Real information for the solver. |
| matrix | The banded matrix being solved (input). |
| rhside | Right hand side. |
| solut | Output solution. |
| nusol | Number of points in solution. |
| nconct | Number of connections in a row of the matrix. |
| infmat | Integer information for the matrix. |
| work | Work array. |
| nwork | Dimension for work array. |
| precon | Preconditioner. |
| nprec | Number of diagonals used in the preconditioner. |
| upperi | Only relevant for computation in periodic domain. |
| loperi | Only relevant for computation in periodic domain. |
| inocnv | Counts occurrence of nonconvergence in solver. |
| itsw | Timestep counter for SWAN. |
| itersw | Iteration counter for SWAN. |

### 5.3.11.13 Subroutine MKPREC

Subroutine MKPREC is used to build a preconditioner.

**Calling Sequence:** mkprec (matrix, nusol, ndimso, nconct, precon, nprec, infmat, mkind)

**Data Declaration:**

| | |
|---|---|
| Real | matrix, precon |
| Integer | nusol, ndimso, nconct, nprec, infmat, mkind |

**Arguments:**

| | |
|---|---|
| matrix | Double precision array in which the matrix of the linear system of equations is stored. In the case of *mkind* = 2 the matrix is scaled. |

| | |
|---|---|
| nusol | The length of the solution vector. |
| ndimso | The dimension of the space for the solver: (*ndimso* = 1 for non-coupled equations, *ndimso* > 1 for coupled equations). |
| nconct | The number of non-zero diagonals of *matrix.* |
| precon | Double precision array in which a preconditioning matrix might be stored, of length *nprec* * *nusol.* It is assumed that *precon* has a similar structure as *matrix.* |
| nprec | Maximum number of diagonals in *precon.* |
| infmat | Array which describes the structure of *matrix.* |
| mkind | The kind of the preconditioner required. |

### 5.3.11.14 Subroutine PREVC

Subroutine PREVC multiplies the vector x with a preconditioner.

**Calling Sequence:** prevc (n, x, b, matrix, ndim, nconct, precon, nprec, infmat, mkind)

**Data Declaration:**

| | |
|---|---|
| Real | x, b, matrix, precon |
| Integer | n, ndim, nconct, nprec, infmat, mkind |

**Arguments:**

| | |
|---|---|
| n | The length of the solution vector. |
| x | The input vector. |
| b | The output vector which is the preconditioner times the vector *x*. |
| matrix | Double precision array in which the matrix of the linear system of equations is stored. |
| ndim | The dimension of the space (*ndim* = 2 or 3). |
| nconct | The number of non-zero diagonals of *matrix.* |
| precon | Double precision array in which a preconditioning matrix might be stored, of length *nprec* * *nusol.* It is assumed that *precon* has a similar structure as *matrix.* |
| nprec | Maximum number of diagonals in *precon.* |
| infmat | Array which describes the structure of the matrix. |
| mkind | The kind of the preconditioner required. |

### 5.3.11.15 Subroutine PRIRES

Subroutine PRIRES prints the norm of the residual.

**Calling Sequence:** prires (text, rnorm, icontr, final)

| | | |
|---|---|---|
| **Data Declaration:** | Real | text, rnorm, final |
| | Integer | icontr |
| **Arguments:** | text | Denotes output form subroutine TEXT. |
| | rnorm | 2-norm of the initial residual. |
| | icontr | Integer array in which information about the solution process must be given by the user. |
| | final | Logical variable telling PRIRES whether this is final iteration or not. |

### 5.3.11.16 Subroutine SWCOVA2D

Subroutine SWCOVA2D computes covariant base vectors in integration points two-dimensional case.

**Calling Sequence:** swcova2d (mxc, myc, xcg, ycg, cva)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xcg, ycg, cva |
| | Integer | mxc, myc |
| **Arguments:** | mxc | Number of points in the x-direction. |
| | myc | Number of points in the y-direction. |
| | xcg | X-coordinates. |
| | ycg | Y-coordinates. |
| | cva | Array containing the covariant basis vectors. |

### 5.3.11.17 Subroutine SWDISDT2

Subroutine SWDISDT2 distributes diffusion terms for transport equation in R2.

**Calling Sequence:** swdisdt2 (mxc, myc, depth, depmin, alphad, matrix, dtsum)

| | | |
|---|---|---|
| **Data Declaration:** | Real | depth, depmin, dtsum, matrix |
| | Integer | mxc, myc, alphad |
| **Arguments:** | mxc | Number of points in the x-direction. |
| | myc | Number of points in the y-direction. |
| | depth | Depth direct addressed. |
| | depmin | Minimum possible depth. |
| | alphad | Direction index of integration. |
| | matrix | Matrix. |
| | dtsum | Derivative contributions to the matrix. |

### 5.3.11.18 Subroutine SWESSBC

Subroutine SWESSBC puts essential boundary conditions into the matrix.

**Calling Sequence:** swessbc (mxc, myc, matrix, rhside, setup)

| | | |
|---|---|---|
| **Data Declaration:** | Real | matrix, rhside, setup |
| | Integer | mxc, myc |

| | | |
|---|---|---|
| **Arguments:** | mxc | Number of points in the x-direction. |
| | myc | Number of points in the y-direction. |
| | matrix | Matrix. |
| | rhside | Right-hand side. |
| | setup | Unknown to be computed direct addressed. |

### 5.3.11.19 Subroutine SWJCTA2D

Subroutine SWJCTA2D computes sqrt(g) x contra-variant base vectors in integration point two-dimensional case.

**Calling Sequence:** swjcta2d (mxc, myc, cva, jcta)

| | | |
|---|---|---|
| **Data Declaration:** | Real | cva, jcta |
| | Integer | mxc, myc |

| | | |
|---|---|---|
| **Arguments:** | mxc | Number of points in the x-direction. |
| | myc | Number of points in the y-direction. |
| | cva | Array containing the covariant basis vectors. |
| | jcta | Jacobian times contra-variant basis vectors:<br>In point pnttyp = 1 base vector 1;<br>In point pnttyp = 2 base vector 2. |

### 5.3.11.20 Subroutine SWSOLV

Subroutine SSWSOLV prepares for ISSOLV.

**Calling Sequence:** swsolv (matrix, rhside, setup, npoint, work, nwork, itsw, iter, upperi, loperi)

| | | |
|---|---|---|
| **Data Declaration:** | Real | matrix, rhside, setup, work, upperi, loperi |
| | Integer | npoint, nwork, itsw, iter |

| | | |
|---|---|---|
| **Arguments:** | matrix | Matrix. |
| | rhside | Right-hand side. |
| | setup | Unknown to be computed direct addressed. |
| | npoint | Number of points *mxc*myc*. |
| | work | Work array. |
| | nwork | Dimension for work array. |
| | itsw | Timestep number. |
| | iter | Iteration number for SWAN. |
| | upperi | Only relevant for computation in periodic domain. |
| | loperi | Only relevant for computation in periodic domain. |

### 5.3.11.21 Subroutine SWTRAD2D

Subroutine SWTRAD2D computes the contribution of diffusion term in R2 for a transport equation per integration point.

**Calling Sequence:** swtrad2d (mxc, myc, wfrcx, wfrcy, depmin, alphad, depth, cva, jcta, cva, jcta, cvc, ctc, dtsum, rhside)

| | | |
|---|---|---|
| **Data Declaration:** | Real | wfrcx, wfrcy, depmin, depth, cva, jcta, cvc, ctc, dtsum, rhside |
| | Integer | mxc, myc, alphad |

| | | |
|---|---|---|
| **Arguments:** | mxc | Number of points in the x-direction. |
| | myc | Number of points in the y-direction. |
| | wfrcx | Force x-component direct addressed. |
| | wfrcy | Force y-component direct addressed. |
| | depmin | Minimum depth. |
| | alphad | Direction index of integration. |
| | depth | Depth direct addressed. |
| | cva | Array containing the covariant basis vectors. |
| | jcta | Jacobian times contra-variant basis vectors<br>In point pnttyp = 1 base vector 1;<br>In point pnttyp = 2 base vector 2. |
| | cvc | Work array containing the covariant WESBEEK vectors. |
| | ctc | Work array containing the contra-variant WESBEEK vectors. |
| | dtsum | Derivative contributions to the matrix. |
| | rhside | Right-hand side. |

### 5.3.11.22 Subroutine VULMAT

| | | |
|---|---|---|
| **Calling Sequence:** | vulmat (n, nconct, a, infmat, upperi, loperi ) | |
| **Data Declaration:** | Real | a, upperi, loperi |
| | Integer | n, nconct, infmat |
| **Arguments:** | n | The length of the solution vector. |
| | nconct | The number of non-zero diagonals of *matrix.* |
| | a | Banded matrix being tested. |
| | infmat | Array which describes the structure of the matrix. |
| | upperi | Only relevant for computation in periodic domain. |
| | loperi | Only relevant for computation in periodic domain. |

### 5.3.11.23 Subroutine VULMT1

| | | |
|---|---|---|
| **Calling Sequence:** | vulmt1 (ntot, band, upperi, loperi, rhv, imatra, imatla, imatda, imatua, imat5l, imat6u, sector, mdc, msc, iddlow, iddtop, isstop, idcmin, idcmax, anybin, idtot, kcgrd, icmax) | |
| **Data Declaration:** | Real | ban, upperi, loperi, rhv, imatra, imatla, imatda, imatua, imat5l, imat6u, sector |
| | Integer | ntot, mdc, msc, iddlow, iddtop, isstop, idcmin, idcmax, idtot, dcgrd, icmax |
| | Logical | anybin |
| **Arguments:** | ntot | Number of points in solution. |
| | band | Banded matrix. |
| | upperi | Only relevant for computation in periodic domain. |
| | loperi | Only relevant for computation in periodic domain. |
| | rhv | RHS of set of equations. |
| | imatra | Coefficients of right hand side of matrix. |
| | imatla | Coefficients of lower diagonal of matrix. |
| | imatda | Coefficients of diagonal of matrix. |
| | imatua | Coefficients of upper diagonal of matrix. |
| | imat5l | Coefficient of lower diagonal in presence of a current. |
| | imat6u | Coefficient of upper diagonal in presence of a current. |
| | sector | The integer array SECTOR denotes which case is present for a certain frequency:<br>= 0: No bins belongs to first sweep, no sector lies within the first sweep<br>= 2: Circle has 2 intersections with sector boundary |

| | |
|---|---|
| | = 4: Circle has 4 intersections with sector boundary |
| | = 1: Full circle lies within the first quadrant, all directions have to taken into account |
| mdc | Maximum counter of directional distribution in computational model. |
| msc | Maximum counter of relative frequency in computational model. |
| iddlow | Minimum counter per sweep taken over all frequencies. |
| iddtop | Maximum counter per sweep taken over all frequencies. |
| isstop | Maximum frequency counter for wave components that are propagated within a sweep. |
| idcmin | Integer array containing minimum counter. |
| idcmax | Integer array containing maximum counter. |
| anybin | Set a particular bin True or False depending on *sector*. |
| idtot | Maximum range between the counters in directional space. |
| kcgrd | Grid address of points of computational stencil. |
| icmax | Maximum array size for the points of a molecule. |

### *5.3.12 SWAN Main Program and Miscellaneous Routines (swanmain FOR File)*

#### 5.3.12.1 Subroutine ERRCHK

Subroutine ERRCHK checks all possible combinations of physical processes if they are being activated and it changes the value of settings if necessary.

**Calling Sequence:** errchk (pool)

**Data Declaration:** Integer pool

**Arguments:** pool Dynamic data pool.

#### 5.3.12.2 Subroutine FLFILE

Subroutine FLFILE updates boundary conditions and non-stationary input fields.

**Calling Sequence:** flfile (icr1, icr2, vnam1, vnam2, jx1, jx2, jx3, jy1, jy2, jy3, cosfc, sinfc, pool, rpool, compda, xcgrid, ycgrid, kgrpnt, ierr)

| | | |
|---|---|---|
| **Data Declaration:** | Real | cosfc, sinfc, rpool, compda, xcgrid, ycgrid |
| | Integer | icr1, igr2, jx1, jx2, jx3, jy1, jy2, jy3, pool, kgrpnt, ierr |
| | Character | vnam1, vnam2 |
| **Arguments:** | icr1 | Location in array *compda* for interpolated input field data (x-comp). |
| | icr2 | Location in array *compda* for interpolated input field data (y-comp) for a scalar field *igr2* = 0. |
| | vnam1 | Pointer name of *pool* array holding values read from file (x-comp). |
| | vnam2 | Pointer name of *pool* array holding values read from file (y-comp). |
| | jx1, jx2, jx3 | Location in array *compda* for interpolated input field data (x-comp). |
| | jy1, jy2, jy3 | Location in array *compda* for interpolated input field data (y-comp). |
| | cosfc | Cosine of the angle between the input and computational grids. |
| | sinfc | Sine of the angle between the input grid and computational grid. |
| | pool | Dynamic data pool. |
| | rpool | Real equivalence for integer *pool.* |
| | compda | Array holding values for computational grid points. |
| | xcgrid | X-coordinate of computational grid points. |
| | ycgrid | Y-coordinate of computational grid points. |
| | kgrpnt | Indirect addresses of computational grid points. |
| | ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |

### 5.3.12.3 Subroutine RBFILE

Subroutine RBFILE reads boundary spectra from one file and additional information of the heading lines.

**Calling Sequence:** rbfile (spcsig, spcdir, bfiled, bsploc, bspdir, rbsdir, bspfrq, rbsfrq, bspecs, bspaux, rbsaux, xytst)

| | | |
|---|---|---|
| **Data Declaration:** | Real | spcdir, spcsig, bspecs, rbsaux, rbsdir, rbsfrq |
| | Integer | bspaux, bspdir, bspfrq, bfiled, bsploc, xytst |
| **Arguments:** | spcsig | Relative frequencies in the computational domain in sigma space. |

| | | |
|---|---|---|
| spcdir | (*,1) | Spectral directions (radians); |
| | (*,2) | Cosine of spectral directions; |
| | (*,3) | Sine of spectral directions; |
| | (*,4) | Cosine^2 of spectral directions; |
| | (*,5) | Cosine*sine of spectral directions; |
| | (*,6) | Sine^2 of spectral directions. |
| bfiled | | Data concerning boundary condition files. |
| bsploc | | Place in array *bspecs* for storing interpolated spectra. |
| bspdir | | Spectral directions of input spectrum. |
| rbsdir | | Real equivalence of *bspdir.* |
| bspfrq | | Spectral frequencies of input spectrum. |
| rbsfrq | | Real equivalence of *bspfrq.* |
| bspecs | | Boundary spectra. |
| bspaux | | Auxiliary array used for interpolation. |
| rbsaux | | Real equivalence of *bspaux.* |
| xytst | | Test points. |

### 5.3.12.4 Subroutine RESPEC

Subroutine RESPEC reads one 1-D or 2-D boundary spectrum from file, and transforms to internal SWAN spectral resolution.

**Calling Sequence:** respec (btype, ndsd, bfiled, unform, dorder, baux1, baux2, baux3, baux4, spcsig, spcdir, bspfrq, bspdir, lspec, ufac, ierr)

**Data Declaration:**

| | | |
|---|---|---|
| Real | spcsig, spcdir, baux1, baux2, baux3, baux4, bspfrq, bspdir, lspec, ufac | |
| Integer | ndsd, bfiled, integer, ierr | |
| Character | btype | |
| Logical | unform | |

**Arguments:**

| | | |
|---|---|---|
| btype | | Type of input. |
| ndsd | | Unit reference number of input file. |
| bfiled | | Options for reading boundary condition file. |
| unform | | If true, unformatted reading is called. |
| dorder | | If < 0, order of directions must be reversed. |
| baux1, baux2, baux3, baux4 | | Auxiliary array. |
| spcsig | | Relative frequencies in computational domain in sigma space. |
| spcdir | (*,1) | Spectral directions (radians); |
| | (*,2) | Cosine of spectral directions; |
| | (*,3) | Sine of spectral directions; |

| | |
|---|---|
| | (*,4) Cosine^2 of spectral directions; |
| | (*,5) Cosine*sine of spectral directions; |
| | (*,6) Sine^2 of spectral directions. |
| bspfrq | Spectral frequencies of input spectrum. |
| bspdir | Spectral directions of input spectrum. |
| lspec | Interpolated spectrum. |
| ufac | Factor used to multiply data. |
| ierr | Error status: |
| | = 0 No error; |
| | = 9 End of file. |

### 5.3.12.5 Subroutine SINARR

Subroutine SINARR calculates energy density at boundary point (x, y, sigma, theta).

**Calling Sequence:** sinarr (pool)

**Data Declaration:** Integer pool

**Arguments:** pool Dynamic data pool.

### 5.3.12.6 Logical Function SINBTG

Subroutine SINBTG checks whether a point given in problem coordinates is in the bottom grid (SINBTG = True) or not (SINBTG = False).

**Calling Sequence:** sinbtg (xp, yp)

**Data Declaration:** Real xp, yp

**Arguments:**

| | |
|---|---|
| xp | X-coordinate (problem grid) of the point. |
| yp | Y-coordinate (problem grid) of the point. |

### 5.3.12.7 Logical Function SINCMP

Subroutine SINCMP checks whether a point given in problem coordinates is in the computational grid (SINCMP = True) or not (SINCMP = False).

**Calling Sequence:** sincmp (xp, yp, xcgrid, ycgrid, kgrpnt, kgrbnd)

**Data Declaration:**

| | |
|---|---|
| Real | xp, yp, xcgrid, ycgrid |
| Integer | kgrpnt, kgrbnd |

| | | |
|---|---|---|
| **Arguments:** | xp | X-coordinate (problem grid) of the point. |
| | yp | Y-coordinate (problem grid) of the point. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |
| | kgrpnt | Grid point addresses. |
| | kgrbnd | Describes computational grid boundary. |

### 5.3.12.8 Subroutine SINUPT

Subroutine SINUPT checks whether the point *xp, yp* (given in problem coordinates) of the output point-set *sname* is located in the computational grid and bottom grid or not. If not, a warning is generated.

**Calling Sequence:** sinupt (psname, xp, yp, xcgrid, ycgrid, kgrpnt, kgrbnd)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xp, yp, xcgrid, ycgrid |
| | Integer | kgrpnt, kgrbnd |
| | Character | psname |

| | | |
|---|---|---|
| **Arguments:** | psname | Name of the output point-set (any type). |
| | xp | X-coordinate of the point (problem coordinates). |
| | yp | Y-coordinate of the point (problem coordinates). |
| | xcgrid | X-coordinate of computational grid in x direction. |
| | ycgrid | Y-coordinate of computational grid in y direction. |
| | kgrpnt | Addresses of the computational grid points. |
| | kgrbnd | Describes the computational grid boundary. |

### 5.3.12.9 Subroutine SNEXTI

**Calling Sequence:** snexti (pool, rpool, bfiles, bsploc, bspdir, rbsdir, bspfrq, rbsfrq, bspaux, rbsaux, bspecs, bgridp, compda, ac1, ac2, spcsig, spcdir, xcgrid, ycgrid, kgrpnt, xytst)

| | | |
|---|---|---|
| **Data Declaration:** | Real | rpool, ac1, ac2, bspecs, compda, rbsaux, rbsdir, rbsfrq, spcdir, spcsig, xcgrid, ycgrid |
| | Integer | pool, bfiles, bsploc, bspdir, bspfrq, bgridp, bspaux, xytst, kgrpnt |

| | | |
|---|---|---|
| **Arguments:** | pool | Data pool. |
| | rpool | Real equivalence of data *pool.* |
| | bfiles | Parameters for reading boundary files. |
| | bsploc | Location where to put boundary values. |

| | |
|---|---|
| bspdir | Spectral directions of boundary spectra. |
| rbsdir | Spectral directions of boundary spectra. |
| bspfrq | Spectral frequencies of boundary spectra. |
| rbsfrq | Spectral frequencies of boundary spectra. |
| bspaux | Auxiliary array used for interpolation. |
| rbsaux | Auxiliary data for interpolation of spectra. |
| bspecs | Boundary spectra. |
| bgridp | Data for interpolating to computational grid points. |
| compda | Values on computational grid. |
| ac1 | Action density spectra on old time level. |
| ac2 | Action density spectra on new time level. |
| spcsig | Relative frequencies in computational domain in sigma space. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| xcgrid | X-coordinate of computational grid in x-direction. |
| ycgrid | Y-coordinate of computational grid in y-direction. |
| kgrpnt | Computational grid point addresses. |
| xytst | Test points. |

### 5.3.12.10 Subroutine SPRCON

Subroutine SPRCON executes some tests on the given model description.

**Calling Sequence:** sprcon (outps, xcgrid, ycgrid, kgrpnt, kgrbnd)

**Data Declaration:**

| | |
|---|---|
| Real | xcgrid, ycgrid |
| Integer | kgrpnt, kgrbnd, outps |

**Arguments:**

| | |
|---|---|
| outps | Contains information about output points. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |
| kgrpnt | Grid point addresses. |
| kgrbnd | Describes the computational grid boundary. |

### 5.3.12.11 Real Function SVALQI

Subroutine SVALQI determines the value of a quantity, such as depth, from an input grid and the current velocity components for point given in problem coordinates.

| | | |
|---|---|---|
| **Calling Sequence:** | svalqi (xp, yp, igrid, arrinp, zero, ixc, iyc) | |
| **Data Declaration:** | Real | xp, yp, arrinp |
| | Integer | igrid, ixc, iyc, zero |
| **Arguments:** | xp | X-coordinate in the computational grid point. |
| | yp | Y-coordinate in the computational grid point. |
| | igrid | Grid indicator. |
| | arrinp | Array holding the values at the input grid locations. |
| | zero | If *zero* = 0, then value outside the grid is zero, otherwise the value is extrapolated. |
| | ixc | Counter for X-coordinate in computational grid (used in curvilinear case). |
| | iyc | Counter for Y-coordinate in computational grid (used in curvilinear case). |

### 5.3.12.12 Program SWAN

Subroutine SWAN is the main program that initializes data pool, and makes common areas empty.

**Common Blocks:** NAMES
TESTDA
OUTPDA
REFNRS
LEESDA
LEESDN
SWNAME
SWGRID
SWCOMG
SWNUMS
SWTEST
SWUITV
SWFYSP
COMPDA

### 5.3.12.14 Subroutine SWINCO

Subroutine SWINCO imposes wave initial conditions at a computational grid.

**Calling Sequence:** swinco (ac2, compda, xcgrid, ycgrid, kgrpnt, spcdir, spcsig, xytst)

| | | |
|---|---|---|
| **Data Declaration:** | Real | ac2, compda, xcgrid, ycgrid, spcdir, spcsig, kgrpnt |
| | Integer | xytst |
| **Arguments:** | ac2 | Action density spectra. |
| | compda | Quantities in grid points. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |
| | kgrpnt | Indirect addresses of grid points. |
| | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| | spcsig | Relative frequencies in the computational domain in sigma space. |
| | xytst | Test points. |

### 5.3.12.15 Subroutine SWINIT

Subroutine SWINIT initializes the dynamic data pool and assigns initial values to the variables in the common blocks.

**Calling Sequence:** swinit (pool, inerr)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | pool, inerr |
| **Arguments:** | pool | Dynamic data pool. |
| | inerr | Number of the initialization error. |

### 5.3.12.16 Subroutine SWMAIN

Subroutine SWMAIN calls subroutines SWINIT, SWREAD, SWCOMP and SWOUTP.

**Calling Sequence:** swmain (pool, rpool, lpool, inerr)

| | | |
|---|---|---|
| **Data Declaration:** | Real | rpool |
| | Integer | pool, inerr |
| | Logical | lpool |
| **Arguments:** | pool | Dynamic data pool. |
| | rpool | Real equivalence to *pool.* |

| | |
|---|---|
| lpool | Logical equivalence to *pool*. |
| inerr | Number of the initialization error. |

### 5.3.12.17 Subroutine SWPREP

Subroutine SWPREP computes the transformation coefficients between the different grids.

**Calling Sequence:** swprep (outda, xcgrid, ycgrid, kgrpnt, obsta, cross, kgrbnd)

**Data Declaration:**

| | |
|---|---|
| Real | xcgrid, ycgrid |
| Integer | outda, kgrpnt, cross, obsta, kgrbnd |

**Arguments:**

| | |
|---|---|
| outda | Contains output data. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |
| kgrpnt | Grid point addresses. |
| kgrbnd | Describes the computational grid boundary. |
| obsta | Array of obstacle parameters. |
| cross | Array which contains 0's if there is no obstacle crossing if an obstacle is crossing between the central point and its neighbor *cross* is equal to the number of the obstacle. |

### 5.3.12.18 Subroutine SWRBC

Subroutine SWRBC determines and writes the depths and currents at a line in the computational grid to a file with reference number *nref*.

**Calling Sequence:** swrbc (pool, rpool, compda, kgrpnt, xcgrid, ycgrid)

**Data Declaration:**

| | |
|---|---|
| Integer | kgrpnt, pool |
| Real | rpool, compda, xcgrid, ycgrid |

**Arguments:**

| | |
|---|---|
| pool | Dynamic data pool. |
| rpool | Real equivalence of *pool*. |
| compda | Values on the computational grid. |
| kgrpnt | Grid point addresses. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |

### *5.3.13 Main Output Routines (swanout1 FOR File)*

#### 5.3.13.1 Subroutine SWIPOL

Subroutine SWIPOL interpolates *finp* to the point given by the computational grid coordinates *xc* and *yc*. The result appears in array *foutp*.

**Calling Sequence:** swipol (finp, excval, xc, yc, mip, foutp, kgrpnt, dep2)

**Data Declaration:**

| | | |
|---|---|---|
| | Real | finp, excval, xc, yc, foutp, dep2 |
| | Integer | mip, kgrpnt |

**Arguments:**

| | | |
|---|---|---|
| | finp | Array of function values defined on the computational grid. |
| | excval | Exception value (assigned if point is outside the computational grid). |
| | xc, yc | Array containing the computational grid coordinates of output points. |
| | mip | Number of output points. |
| | foutp | Array of interpolated values for the output points. |
| | kgrpnt | Index for indirect addressing. |
| | dep2 | Depth at the computational grid points. |

#### 5.3.13.2 Subroutine SWODDC

Subroutine SWODDC decodes output point set data.

**Calling Sequence:** swoddc (outps, psname, pstype, mip, mxk, myk, xnlen, ynlen, mxn, myn, xpcn, ypcn, alpcn, xcgrid, ycgrid, rtype)

**Data Declaration:**

| | | |
|---|---|---|
| | Real | xcgrid, ycgrid, xnlen, ynlen, xpcn, ypcn, alpcn |
| | Integer | outps, mip, mxk, myk, mxn, myn |
| | Character | psname, pstype, rtype |

**Arguments:**

| | | |
|---|---|---|
| | outps | Array containing output data. |
| | psname | Name of output point set referred to. |
| | pstype | Type of output point set. |
| | mip | Number of output points. |
| | mxk | Number of output points in X-direction (Frame). |
| | myk | Number of output points in Y-direction (Frame). |
| | xnlen, ynlen | (X, Y) length of the nested grid. |
| | mxn, myn | Number of meshes in X, Y direction for the nested grid. |
| | xpcn, ypcn | Location of the origin of the nested grid. |

| | |
|---|---|
| alpcn | Angle of the nested grid with the positive x-axis, counter-clockwise measured. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |
| rtype | Indicates type of output; "PLOT" means that a spatial plot is made. |

### 5.3.13.3 Subroutine SWOEXA

Subroutine SWOEXA calculates quantities for which the spectral action density is necessary.

**Calling Sequence:** swoexa (oqproc, bkc, mip, xc, yc, voqr, voq, ac2, acloc, spcsig, wk, cg, spcdir, ne, ned, kgrpnt, depxy)

**Data Declaration:**

| | |
|---|---|
| Real | xc, yc, voq, ac2, spcsig, spcdir, wk, cg, ne, ned, depxy, acloc |
| Integer | mip, voqr, kgrpnt |
| Logical | oqproc |

**Arguments:**

| | |
|---|---|
| oqproc | Processing of output quantities. |
| bkc | Variable used to flag variables for calculation for purpose of writing to output. |
| mip | Number of output points. |
| xc, yc | Computational grid coordinates. |
| voqr | Location in *voq* of certain output quantities. |
| voq | Values of output quantities. |
| ac2 | Action densities. |
| acloc | Local action density spectrum. |
| spcsig | Relative frequencies in the computational domain in sigma space. |
| wk | Wave number in output point. |
| cg | Group velocity in output point. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| ne | Ratio of group and phase velocity. |
| ned | Derivative of *ne* with respect to depth. |
| kgrpnt | Index for indirect addressing. |
| depxy | Depth in points of the computational grid. |

### 5.3.13.4 Subroutine SWOEXC

Subroutine SWOEXC calculates the computational grid coordinates of the output points.

**Calling Sequence:** swoexc (outps, pstype, mip, xp, yp, xc, yc, kgrpnt, xcgrid, ycgrid, kgrbnd)

**Data Declaration:**

| | | |
|---|---|---|
| | Real | xp, yp, xc, yc, xcgrid, ycgrid |
| | Integer | outps, mip, kgrpnt, kgrbnd |
| | Character | pstype |

**Arguments:**

| | | |
|---|---|---|
| | outps | Array containing output data. |
| | pstype | Type of output point set. |
| | mip | Number of output points. |
| | xp, yp | User coordinates of output point. |
| | xc, yc | Computational grid coordinates. |
| | kgrpnt | Index for indirect addressing. |
| | xcgrid | X-coordinate of computational grid in x direction. |
| | ycgrid | Y-coordinate of computational grid in y direction. |
| | kgrbnd | Describes the computational grid boundary. |

### 5.3.13.5 Subroutine SWOEXD

Subroutine SWOEXC calculates the distance, depth, Ux, Uy, etc.

**Calling Sequence:** swoexd (oqproc, mip, xc, yc, voqr, voq, compda, kgrpnt)

**Data Declaration:**

| | | |
|---|---|---|
| | Real | xc, yc, compda, voq |
| | Integer | mip, kgrpnt, voqr |
| | Logical | oqproc |

**Arguments:**

| | | |
|---|---|---|
| | oqproc | Y/n process output quantities. |
| | mip | Number of output points. |
| | xc, yc | Computational grid coordinates. |
| | voqr | Location in *voq* of certain output quantities. |
| | voq | Values of output quantities. |
| | compda | Array holding values for computational grid points. |
| | kgrpnt | Index for indirect addressing. |

### 5.3.13.6 Subroutine SWOEXF

Subroutine SWOEXF calculates wave-driven force (output quantity IVTYPE = 20).

| | | |
|---|---|---|
| **Calling Sequence:** | swoexf (mip, xc, yc, voqr, voq, ac2, dep2, spcsig, wk, cg, spcdir, ne, ned, kgrpnt, xcgrid, ycgrid) | |
| **Data Declaration:** | Real | xc, yc, voq, ac2, dep2, spcsig, wk, cg, spcdir, ne, ned, xcgrid, ycgrid |
| | Integer | mip, voqr, kgrpnt |
| **Arguments:** | mip | Number of output points. |
| | xc, yc | Computational grid coordinates of output point. |
| | voqr | Location in *voq* of a certain output quantity. |
| | voq | Values of output quantities. |
| | ac2 | Action density. |
| | dep2 | Depth at the computational grid points. |
| | spcsig | Relative frequencies in the computational domain in sigma space. |
| | wk | Wave number in output point. |
| | cg | Group velocity in output point. |
| | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| | ne | Ratio of group and phase velocity. |
| | ned | Derivative of *ne* with respect to depth. |
| | kgrpnt | Index for indirect addressing. |
| | xcgrid | X-coordinate of computational grid in x direction. |
| | ycgrid | Y-coordinate of computational grid in y direction. |

### 5.3.13.7 Subroutine SWOINA

Subroutine SWOINA interpolates local action density *acloc* from array *ac2*.

| | | |
|---|---|---|
| **Calling Sequence:** | swoina (xc, yc, ac2, acloc, kgrpnt, depxy) | |
| **Data Declaration:** | Real | xc, yc, ac2, acloc, depxy |
| | Integer | kgrpnt |
| **Arguments:** | xc, yc | Computational grid coordinates. |
| | ac2 | Action densities. |
| | acloc | Local action density spectrum. |
| | kgrpnt | Index for indirect addressing. |
| | depxy | Depth in points of the computational grid. |

### 5.3.13.8 Subroutine SWORDC

Subroutine SWORDC decodes output requests.

**Calling Sequence:** swordc (outi, outr, rtype, psname, nvoqp, oqproc, bkc, voqr, logact)

**Data Declaration:**

| | | |
|---|---|---|
| | Integer | voqr, bkc, outi, nvoqp |
| | Real | outr |
| | Logical | oqproc, logact |
| | Character | rtype, psname |

**Arguments:**

| | | |
|---|---|---|
| | outi | Array for storage of information regarding location, type of output. |
| | outr | Code for one output request. |
| | rtype | Type of output. |
| | psname | Name of output point set referred to. |
| | nvoqp | Number of data per output point. |
| | oqproc | Whether or not an output quantity must be processed. |
| | bkc | Variable used to flag variables for calculation for purpose of writing to output. |
| | voqr | Place of each output quantity. |
| | logact | Logical variable; TRUE enables output. |

### 5.3.13.9 Subroutine SWOUTP

Subroutine SWOUTP processes the output requests.

**Calling Sequence:** swoutp (outda, routda, loutda, ac2, spcsig, spcdir, compda, xytst, kgrpnt, xcgrid, ycgrid, kgrbnd)

**Data Declaration:**

| | | |
|---|---|---|
| | Real | routda, ac2, spcsig, spcdir, xcgrid, ycgrid, compda |
| | Integer | outda, xytst, kgrpnt, kgrbnd |
| | Logical | loutda |

**Arguments:**

| | | |
|---|---|---|
| | outda | Array containing output data, requests. |
| | routda | Real equivalence of *outda*. |
| | loutda | Logical equivalence of *outda*. |
| | ac2 | Action density in all computational points. |
| | spcsig | Relative frequencies in the computational domain in sigma space. |
| | spcdir | Spectral directions, cosines and sines. |
| | compda | Array holding values for the computational grid points. |

| | |
|---|---|
| xytst | Test points. |
| kgrpnt | Index for indirect addressing. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |
| kgrbnd | Describes the computational grid boundary. |

### *5.3.14 Output Routines (swanout2 FOR File)*

#### 5.3.14.1 Subroutine PLOTCG

Subroutine PLOTCG plots the computational grid.

**Calling Sequence:** plotcg (ixmax, iymax, ixmin, iymin, lincol, cx, cy, kgrpnt)

**Data Declaration:**

| | |
|---|---|
| Integer | ixmax, iymax, ixmin, iymin, lincol, kgrpnt |
| Real | cx, cy |

**Arguments:**

| | |
|---|---|
| ixmax | Maximum X for which computational grid is to be plotted. |
| iymax | Maximum Y for which computational grid is to be plotted. |
| ixmin | Minimum X for which computational grid is to be plotted. |
| iymin | Minimum Y for which computational grid is to be plotted. |
| lincol | Line color (pen number) used for plotting. |
| cx, cy | Coordinates of computational grid points. |
| kgrpnt | Array grid point indices. |

#### 5.3.14.2 Subroutine SBLKPT

Subroutine SBLKPT writes the block output either on paper or to data file.

**Calling Sequence:** sblkpt (ipd, nref, dfac, psname, qunit, mxk, myk, idla, string, oqvals)

**Data Declaration:**

| | |
|---|---|
| Real | dfac, oqvals |
| Integer | ipd, nref, mxk, myk, idla |
| Character | psname, qunit, string |

**Arguments:**

| | |
|---|---|
| ipd | Switch for printing on paper (*ipd* = 1) or writing to data file (*ipd* = 2 or 3). |

| | |
|---|---|
| nref | Unit reference number of output file. |
| dfac | Multiplication factor of block output. |
| psname | Name of output point set (frame). |
| qunit | Physical unit (dimension) of variable. |
| mxk | Number of points in x-direction of frame. |
| myk | Number of points in y-direction of frame. |
| idla | Controls layout of output. |
| string | Description of output variable. |
| oqvals | Generic array containing variable which is being written. |

### 5.3.14.3 Subroutine SPLOER

Subroutine SPLOER draws a plot with the locations of error points.

**Calling Sequence:** sploer (oreq, xcgrid, ycgrid)

**Data Declaration:**

| | |
|---|---|
| Real | xcgrid, ycgrid |
| Integer | oreq |

**Arguments:**

| | |
|---|---|
| oreq | Array containing output requests. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |

### 5.3.14.4 Character Function SUHEAD

Subroutine SUHEAD prepares a unit for the table print output in the form: [unit].

**Calling Sequence:** suhead (qunit)

**Data Declaration:** Character qunit

**Arguments:**

| | |
|---|---|
| qunit | Unit of the variable to be printed in the table headings. |

### 5.3.14.5 Subroutine SWBLOK

Subroutine SWBLOK prepares output in the form of a block that is printed by subroutine SBLKPT.

**Calling Sequence:** swblok (rtype, oreq, psname, mxk, myk, voqr, voq)

| | | |
|---|---|---|
| **Data Declaration:** | Real | voq |
| | Integer | voqr, oreq, mxk, myk |
| | Character | rtype, psname |

| | | |
|---|---|---|
| **Arguments:** | rtype | Type of output request:<br>BLKP for output on paper;<br>BLKD and BLKL for output to data file. |
| | oreq | Array containing current output request. |
| | psname | Name of output frame. |
| | mxk | Number of grid points in x-direction. |
| | myk | Number of grid points in y-direction. |
| | voqr | Gives location in array *voq* where to find a variable. |
| | voq | Values of variables for all output points. |

### 5.3.14.6 Subroutine SWCMSP

Subroutine SWCMSP computes energy density spectrum 1-D or 2-D.

**Calling Sequence:** swcmsp (otype, xc, yc, ac2, acloc, spcsig, dep, dep2, ux, uy, ecos, esin, ofac, kgrpnt, ierr)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xc, yc, ac2, acloc, spcsig, dep, dep2, ux, uy, ecos, esin, ofac |
| | Integer | otype, kgrpnt, ierr |

| | | |
|---|---|---|
| **Arguments:** | otype | Type of spectrum wanted: 2 or -2 for 2-D spectrum, 1 or -1 for 1-D frequency spectrum positive: relative frequency negative: absolute frequency. |
| | xc, yc | Coordinates of output location(s). |
| | ac2 | Action densities. |
| | acloc | $\|otype\| = 2$: 2-D spectrum at one output location.<br>$\|otype\| = 1$: 1-D spectra at output locations. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | dep | Depths at output location. |
| | dep2 | Depth. |
| | ux, uy | Current velocities at output location. |
| | ecos | Cosines of spectral directions. |
| | esin | Sines of spectral directions. |
| | ofac | Output factor (if *inrhog* = 1, equal to *rho*grav*). |
| | kgrpnt | Array grid point indices. |

| | |
|---|---|
| ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |

### 5.3.14.7 Subroutine SWPLOT

Subroutine SWPLOT prepares to plot contour lines and vector patterns.

**Calling Sequence:** swplot (oreq, mxk, myk, ppname, voqr, voq, orer, places, placer, clines, cliner, psdata, outpr, xcgrid, ycgrid, kgrpnt, kgrbnd, i_voq)

**Data Declaration:**

| | |
|---|---|
| Real | cliner, outpr, placer, xcgrid, ycgrid, orer |
| Integer | i_voq, mxk, myk, voqr, clines, ppname, oreq, places, psdata, kgrpnt, kgrbnd |

**Arguments:**

| | |
|---|---|
| oreq | Array containing output requests. |
| mxk, myk | Number of grid points of output frame. |
| ppname | Output frame. |
| voqr | Gives location in array *voq* where to find a variable. |
| voq | Values of variables for all output points. |
| orer | Real equivalence of *oreq*. |
| places | Data on town and region names. |
| placer | Real equivalence of *places*. |
| clines | Data on coastlines. |
| cliner | Real equivalence of *clines*. |
| psdata | Data on output point sets. |
| outpr | Real equivalence of *psdata*. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |
| kgrpnt | Array grid point indices. |
| kgrbnd | Describes the computational grid boundary. |
| i_voq | Integer equivalence of *voq*. |

### 5.3.14.8 Subroutine SWSPEC

Subroutine SWSPEC prints action density spectrum in the form of a table.

**Calling Sequence:** swspec (rtype, oreq, mip, voqr, voq, ac2, acloc, spcsig, spcdir, dep2, kgrpnt)

**Data Declaration:**

| | |
|---|---|
| Real | voq, ac2, spcsig, spcdir, dep2 |
| Integer | oreq, mip, voqr, kgrpnt |

| | | |
|---|---|---|
| | Character | rtype |
| **Arguments:** | rtype | Type of output request:<br>*spec* for 2-D spectral output;<br>*spe1* for 1-D frequency spectrum. |
| | oreq | Array containing output request data of request currently being processed. |
| | mip | Number of output points in set *psname*. |
| | voqr | Gives location in array *voq* where to find a variable. |
| | voq | Values of variables for all output points. |
| | ac2 | Action densities. |
| | acloc | Case *spec*: 2-D spectrum at one output location.<br>Case *spe1*: 1-D spectra at output locations. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| | dep2 | Depth. |
| | kgrpnt | Array grid point indices. |

### 5.3.14.9 Subroutine SWSTAR

Subroutine SWSTAR plots directional distribution of action transport.

**Calling Sequence:** swstar (oreq, mxk, myk, voqr, voq, orer, kgrpnt, spcsig, spcdir, ac2, acloc, wavn, cg, ne, ned)

| | | |
|---|---|---|
| **Data Declaration:** | Real | voq, spcsig, spcdir, ac2, acloc, wavn, cg, ne, ned, orer |
| | Integer | oreq, mxk, myk, voqr, kgrpnt |
| **Arguments:** | oreq | Array containing output requests. |
| | mxk, myk | Number of grid points of output frame. |
| | voqr | Gives location in array *voq* where to find a variable. |
| | voq | Values of variables for all output points. |
| | orer | Real equivalence of *oreq*. |
| | kgrpnt | Array grid point indices. |
| | spcsig | Relative frequencies in computational domain in |

| | |
|---|---|
| | sigma space. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| ac2 | Action densities. |
| acloc | Spectral action densities in output point. |
| wavn | Wave numbers. |
| cg | Energy property velocity. |
| ne | Unused. |
| ned | Unused. |

#### 5.3.14.10 Subroutine SWTABP

Subroutine SWTABP prints output in the form of a table for any type of output point set.

**Calling Sequence:** swtabp (rtype, oreq, psname, mip, voqr, voq)

**Data Declaration:**

| | |
|---|---|
| Real | voq |
| Integer | mip, voqr, oreq |
| Character | rtype, psname |

**Arguments:**

| | |
|---|---|
| rtype | Type of output request. |
| oreq | Array containing output requests. |
| psname | Name of output point set (frame). |
| mip | Number of output points in set *psname*. |
| voqr | Gives location in array *voq* where to find a variable. |
| voq | Values of variables for all output points. |

### *5.3.15 Output routines (swanout3 FOR File)*

#### 5.3.15.1 Subroutine PLSPEC

**Calling Sequence:** plspec (oreq, orer, spcsig, logpl, acloc, ip, xc, yc, hsig, aper, pper, adir, dspr, wvx, wvy, spcdir)

**Data Declaration:**

| | |
|---|---|
| Real | orer, spcsig, acloc, xc, yc, hsig, aper, pper, adir, dspr, wvx, wvy, spcdir |
| Integer | oreq, ip |

| | | |
|---|---|---|
| | Logical | logpl |
| **Arguments:** | oreq | Array containing output requests. |
| | orer | Real equivalence of *oreq*. |
| | spcsig | Relative frequencies in the computational domain in sigma space. |
| | logpl | Logical array used as working space with dimension (*nfreq, nangl*). |
| | acloc | \|*otype*\| = 2: 2-D spectrum at one output location.<br>\|*otype*\| = 1: 1-D spectra at output locations. |
| | ip | Point to be plotted. |
| | xc, yc | Coordinates of output location(s). |
| | hsig | Significant wave height. |
| | aper | Average wave period. |
| | pper | Peak wave period. |
| | adir | Average (mean) wave direction. |
| | dspr | One-sided directional width of spectrum. |
| | wvx, wvy | X and y component, respectively, of wind velocity. |
| | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |

### 5.3.15.2 Subroutine PLTAR1

Subroutine PLTAR1 plots an arrow (centered).

**Calling Sequence:** pltar1 (x0, y0, arl, tha, th2, fac2, arlmin)

**Data Declaration:** Real x0, y0, arl, tha, th2, fac2, arlmin

| | | |
|---|---|---|
| **Arguments:** | x0, y0 | Center coordinates of *array*. |
| | arl | Arrow length. |
| | tha | Direction of arrow. |
| | th2 | Angle in head of arrow. |
| | fac2 | Length factor head arrow. |
| | arlmin | Minimum arrow length. |

### 5.3.15.3 Subroutine PLTAR2

Subroutine PLTR2 plots an arrow (centered).

**Calling Sequence:** pltar2 (x0, y0, arl, tha, th2, fac2, arlmin)

**Data Declaration:** Real x0, y0, arl, tha, th2, fac2, arlmin

**Arguments:**

| | |
|---|---|
| x0, y0 | Center coordinates of array. |
| arl | Arrow length. |
| tha | Direction of arrow. |
| th2 | Angle in head of arrow. |
| fac2 | Length factor head arrow. |
| arlmin | Minimum arrow length. |

### 5.3.15.4 Subroutine PLTCIR

Subroutine PLTCIR plots a circle with radius *r* around the origin.

**Calling Sequence:** pltcir (r, dashln)

**Data Declaration:** Real r, dashln

**Arguments:**

| | |
|---|---|
| r | Radius of circle in plot units. |
| dashln | Length of dashes. |

### 5.3.15.5 Subroutine PLTISO

Subroutine PLTISO is a contour plot with isolines on a rectangular grid.

**Calling Sequence:** pltiso (spcsig, chts, logpl, acloc, spcdir)

**Data Declaration:**

| | |
|---|---|
| Real | spcsig, chts, acloc, spcdir |
| Logical | logpl |

**Arguments:**

| | |
|---|---|
| spcsig | Relative frequencies in the computational domain in sigma space. |
| chts | Real array with dimension of at least *nhts*, containing contour heights. |
| logpl | Logical array used as working space with dimension (*nfreq, nangl*). |
| acloc | $\|otype\| = 2$: 2-D spectrum at one output location.<br>$\|otype\| = 1$: 1-D spectra at output locations. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions; |

(*,4) Cosine^2 of spectral directions;
(*,5) Cosine*sine of spectral directions;
(*,6) Sine^2 of spectral directions.

**Common Blocks:** CPLT1

### 5.3.15.6 Subroutine PLTLN1

Subroutine PLTLN1 plots a (dashed) line.

**Calling Sequence:** pltln1 (x1, x2, y1, y2, dashln)

**Data Declaration:** Real x1, x2, y1, y2, dashln

**Arguments:**

| | | |
|---|---|---|
| | x1 | X-coordinate of starting point. |
| | x2 | X-coordinate of end point. |
| | y1 | Y-coordinate of starting point. |
| | y2 | Y-coordinate of end point. |
| | dashln | Length of dashes. |

### 5.3.15.7 Subroutine PLTSEG

Subroutine PLTSEG computes coordinates of the begin and end points of a line starting on a circle with radius *radc* with an end point on the side of a square box with size *psmax*. The direction of the line is *psi* degrees.

**Calling Sequence:** pltseg (radc, psmax, psi, x1, x2, y1, y2)

**Data Declaration:** Real radc, psmax, psi, x1, x2, y1, y2

**Arguments:**

| | | |
|---|---|---|
| | radc | Radius of inner circle. |
| | psmax | Size of outer box. |
| | psi | Direction in degrees. |
| | x1 | X-coordinate of starting point. |
| | x2 | X-coordinate of end point. |
| | y1 | Y-coordinate of starting point. |
| | y2 | Y-coordinate of end point. |

### 5.3.15.8 Subroutine PLT2DS

Subroutine PLT2DS is a polar contour plot of 2-D spectrum.

**Calling Sequence:** plt2ds (norms2, spcsig, rcir, logpl, acloc, nhts, chts, iln, spcdir)

**Data Declaration:**

| | |
|---|---|
| Real | spcsig, spcdir, acloc, rcir, chts |
| Logical | logpl |
| Integer | norms2, nhts, iln |

**Arguments:**

| | |
|---|---|
| norms2 | Parameter specifying normalization. |
| spcsig | Relative frequencies in computational domain in sigma space. |
| rcir | Radii of circles. |
| logpl | Logical array used as working space with dimension (*nfreq, nangl*). |
| acloc | \|*otype*\| = 2: 2-D spectrum at one output location.<br>\|*otype*\| = 1: 1-D spectra at output locations. |
| nhts | Number of contour heights (maximum = 14). |
| chts | Real array with dimension of at least *nhts*, containing contour heights. |
| iln | Parameter specifying whether the lines and circles must be plotted:<br>= 0 no lines and circles;<br>= 1 lines are plotted;<br>= 2 circles are plotted;<br>= 3 lines and circles are plotted. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |

### 5.3.15.9 Subroutine PSIGMA

Subroutine PSIGMA draws a sigma.

**Calling Sequence:** psigma (x, y, dxout3, dyout3)

**Data Declaration:**

| | |
|---|---|
| Real | x, y, dxout3, dyout3 |

**Arguments:**

| | |
|---|---|
| x | X-coordinate of lower left corner. |
| y | Y-coordinate of lower left corner. |
| dxout3 | Size in X-direction. |
| dyout3 | Size in Y-direction. |

### 5.3.15.10 Subroutine PTHETA

Subroutine PTHETA draws a theta.

**Calling Sequence:** ptheta (x, y, dxout3, dyout3)

**Data Declaration:** Real x, y, dxout3, dyout3

**Arguments:**

| | | |
|---|---|---|
| x | X-coordinate of lower left corner. | |
| y | Y-coordinate of lower left corner. | |
| dxout3 | Size in X-direction. | |
| dyout3 | Size in Y-direction. | |

### 5.3.15.11 Subroutine SWPLSP

**Calling Sequence:** swplsp (rtype, oreq, orer, mip, ac2, acloc, aux, laux, voq, voqr, spcsig, spcdir, kgrpnt, dep2)

**Data Declaration:**

| | |
|---|---|
| Integer | oreq, voqr, kgrpnt, mip |
| Real | orer, ac2, acloc, aux, voq, spcsig, spcdir, dep2 |
| Logical | laux |
| Character | rtype |

**Arguments:**

| | |
|---|---|
| rtype | Type of output request:<br>*spec* for 2-D spectral output;<br>*spe1* for 1-D frequency spectrum. |
| oreq | Array containing output requests. |
| orer | Real equivalence of *oreq*. |
| mip | Number of output points in set *psname*. |
| ac2 | Action densities. |
| acloc | \|*otype*\| = 2: 2-D spectrum at one output location;<br>\|*otype*\| = 1: 1-D spectra at output locations. |
| aux | Action density at one location in space. |
| laux | Logical equivalence of *aux*. |
| voqr | Gives location in array *voq* where to find a variable. |
| voq | Values of variables for all output points. |
| spcsig | Relative frequencies in the computational domain in sigma space. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |

| | |
|---|---|
| kgrpnt | Array grid point indices. |
| dep2 | Depth. |

#### 5.3.15.12 Subroutine TRAFO

Subroutine TRAFO transforms polar coordinates to rectangular coordinates.

**Calling Sequence:** trafo (xin, yin, xout, yout, spcdir)

**Data Declaration:** Real xin, yin, xout, yout, spcdir

**Arguments:**

| | |
|---|---|
| xin | (Normalized) frequency. |
| yin | Direction (number of direction steps). |
| xout | Output X. |
| yout | Output Y. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |

### *5.3.16 Preconditioning Subroutines (swanpre1 FOR File)*

#### 5.3.16.1 Subroutine BACKUP

Subroutine BACKUP is a backup current state of the wave field to a file.

**Calling Sequence:** backup (ac2, spcsig, spcdir, kgrpnt, xcgrid, ycgrid)

**Data Declaration:**

| | |
|---|---|
| Real | ac2, spcsig, spcdir, xcgrid, ycgrid |
| Integer | kgrpnt |

**Arguments:**

| | |
|---|---|
| ac2 | Action density as function of D, S, X, Y at time T. |
| spcsig | Relative frequencies in the computational domain in sigma space. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |

| | |
|---|---|
| kgrpnt | Indirect addresses for grid points. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |

#### 5.3.16.2 Subroutine CGBOUN

Subroutine CGBOUN determines an array containing all points of (closed) boundary/boundaries within the computational grid.

**Calling Sequence:** cgboun (kgrpnt, kgrbnd)

**Data Declaration:** Integer kgrpnt, kgrbnd

**Arguments:**

| | |
|---|---|
| kgrpnt | Indirect addresses for grid points. |
| kgrbnd | Array containing all boundary points (+ 2 extra zeros as area separators for all separated areas). |

#### 5.3.16.3 Subroutine CGINIT

Subroutine CGINIT initializes arrays for description of the computational grid.

**Calling Sequence:** cginit (pool, rpool, logcom)

**Data Declaration:**

| | |
|---|---|
| Integer | pool |
| Real | rpool |
| Logical | logcom |

**Arguments:**

| | |
|---|---|
| pool | Dynamic data pool. |
| rpool | Real equivalence of *pool.* |
| logcom | The logical variable *logcom* has a record about which commands have been given to know if all the information for certain command is available. |

#### 5.3.16.4 Subroutine INITVA

Subroutine INITVA processes command INIT and computes the initial state of the wave field.

**Calling Sequence:** initva (ac2, spcsig, edirs, spcdir, kgrpnt, xcgrid, ycgrid, logcom, xytst)

**Data Declaration:** Real spcsig, spcdir, xcgrid, ycgrid, ac2, edirs

| | | |
|---|---|---|
| | Integer | kgrpnt, xytst |
| | Logical | logcom |
| **Arguments:** | ac2 | Action density as function of D, S, X, Y at time T. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | edirs | Not used. |
| | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| | kgrpnt | Indirect addresses for grid points. |
| | xcgrid | X-coordinate of computational grid in x direction. |
| | ycgrid | Y-coordinate of computational grid in y direction. |
| | logcom | The logical variable *logcom* has a record about which commands have been given to know if all the information for certain command is available. |
| | xytst | Test points. |

### 5.3.16.5 Logical Function PVALID

Subroutine PVALID finds whether or not a couple (*ix, iy*) represents a valid grid point.

**Calling Sequence:** pvalid (ix, iy, kgrpnt)

**Data Declaration:** Integer ix, iy, kgrpnt

| | | |
|---|---|---|
| **Arguments:** | ix, iy | X- and y-indices of the point under consideration. |
| | kgrpnt | Indirect addresses for grid points. |

### 5.1.16.6 Subroutine SEPARAREA

Subroutine SEPARAREA separates the areas that could be connected with a one cell connection.

**Calling Sequence:** separarea (ix, iy, kgrpnt, idir)

**Data Declaration:** Integer ix, iy, kgrpnt, idir

| | | |
|---|---|---|
| **Arguments:** | ix, iy | X- and y-indices of point under consideration. |
| | kgrpnt | Indirect addresses for grid points. |
| | idir | Index for direction. |

### 5.3.16.7 Subroutine SINPGR

Subroutine SINPGR reads parameters of an input grid.

**Calling Sequence:** sinpgr (igrid1, igrid2, snameg, outps, xcgrid, ycgrid)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xcgrid, ycgrid, outps |
| | Integer | igrid1, igrid2 |
| | Character | snameg |

| | | |
|---|---|---|
| **Arguments:** | igrid1 | Grid number for which parameters are read. |
| | igrid2 | Grid number for which parameters are read only relevant if > 0. |
| | snameg | Name of output frame corresponding to input grid. |
| | outps | Array storing output frame data. |
| | xcgrid | X-coordinate of computational grid in x direction. |
| | ycgrid | Y-coordinate of computational grid in y direction. |

**Common Blocks:** REFNRS
SWCOMG
SWFYSP
SWGRID
SWTEST
SWUITV
TESTDA
TIMFIL

### 5.3.16.8 Subroutine SREDEP

Subroutine SREDEP reads depths and/or currents.

**Calling Sequence:** sredep (pool, lwindr, lwindm, logcom, rpool)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | pool, lwindr, lwindm |
| | Real | rpool |
| | Logical | logcom |

| | | |
|---|---|---|
| **Arguments:** | pool | Output variable that is filled with computational data needed for the simulation by SWAN. |

| | |
|---|---|
| lwindr | Describes type of wind information being read. |
| lwindm | Describes wind input physics mode. |
| logcom | The logical variable *logcom* has a record about which commands have been given to know if all the information for certain command is available. |
| rpool | Real equivalence of *pool* array. |

#### 5.3.16.9 Subroutine SSFILL

Subroutine SSFILL discretizes in frequency (sigma) and direction (theta).

**Calling Sequence:** ssfill (spcsig, spcdir)

**Data Declaration:** Real spcsig, spcdir

**Arguments:**

| | |
|---|---|
| spcsig | Relative frequencies in computational domain in sigma space. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |

#### 5.3.16.10 Subroutine SWDIM

Subroutine SWDIM computes depths and currents by bilinear interpolation and usually writes to file INSTR.

**Calling Sequence:** swdim (kgrpnt, depth, xcgrid, ycgrid, xytst)

**Data Declaration:**

| | |
|---|---|
| Real | xcgrid, ycgrid, depth |
| Integer | kgrpnt, xytst |

**Arguments:**

| | |
|---|---|
| kgrpnt | Indirect addresses for grid points. |
| depth | The water depth array. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |
| xytst | Test point. |

### 5.3.16.11 Subroutine SWREAD

Subroutine SWREAD reads and processes the user commands describing the model.

**Calling Sequence:** swread (comput, pool, rpool)

**Data Declaration:**

| | |
|---|---|
| Real | rpool |
| Integer | pool |
| Character | comput |

**Arguments:**

| | |
|---|---|
| comput | Output variable that determines the sort of computation to be performed by SWAN:<br>= comp Computation requested;<br>= noco No computation but output requested;<br>= retr Retrieve data from previous computation;<br>= stop Make computation, output and stop. |
| pool | Output variable that is filled with computational data needed for the simulation by SWAN. |
| rpool | Real equivalence for *pool.* |

**Common Blocks:**

CBOUP
COMPDA
LEESDA
LEESDN
NAMES
OUTPDA
REFNRS
SWANWL
SWCOMG
SWFYSP
SWGRID
SWNAME
SWNUMS
SWUITV
SWTEST
TESTDA
TIMCOM
TIMRED
WAMBOU
WFILNM

#### 5.3.16.12 Logical Function VALIDBP

Subroutine VALIDBP checks to see whether or not the point with index (*ix, iy*) can be a valid boundary point.

| | | |
|---|---|---|
| **Calling Sequence:** | validbp (ix, iy, kgrpnt, wnp) | |
| **Data Declaration:** | Integer | ix, iy, kgrpnt, wnp |
| **Arguments:** | ix, iy | X- and y-indices of point under consideration. |
| | kgrpnt | Indirect addresses for grid points. |
| | wnp | Number of wet neighboring points. |

### *5.3.17 File Two of the Preconditioning Subroutines (swanpre2 FOR File)*

#### 5.3.17.1 Subroutine BCFILE

Subroutine BCFILE reads file data for boundary condition.

| | | |
|---|---|---|
| **Calling Sequence:** | bcfile (fbcnam, bctype, bfiled, bsploc, bspdir, rbsdir, bspfrq, rbsfrq, bgridp, bspaux, xcgrid, ycgrid, kgrpnt, xytst, kgrbnd, donall) | |
| **Data Declaration:** | Real | rbsdir, rbsfrq, xcgrid, ycgrid |
| | Integer | bspdir, bspfrq, bfiled, bspaux, kgrpnt, bgridp, bsploc, xytst, kgrbnd |
| | Character | fbcnam, bctype |
| | Logical | donall |
| **Arguments:** | fbcnam | Filename of boundary data file. |
| | bctype | If value is "NEST" $\rightarrow$ nesting b.c. |
| | bfiled | Data concerning boundary condition files. |
| | bsploc | Place in array *bspecs* where to store interpolated spectra. |
| | bspdir | Spectral directions of input spectrum. |
| | rbsdir | Real equivalence of *bspdir*. |
| | bspfrq | Spectral frequencies of input spectrum. |
| | rbsfrq | Real equivalence of *bspfrq*. |
| | bgridp | Data concerning boundary grid points. |
| | bspaux | Auxiliary array used for interpolation. |
| | xcgrid | X-coordinate of computational grid points. |
| | ycgrid | Y-coordinate of computational grid points. |
| | kgrpnt | Indirect addresses of grid points. |
| | xytst | *Ix, iy* of test points. |

| | |
|---|---|
| kgrbnd | Array of boundary grid points. |
| donall | Declares if the nesting boundary is open or closed. *Donall* is defined by the users. |

#### 5.3.17.2 Subroutine BC_POINTS

Subroutine BC_POINTS interpolates grid points to the SWAN computational grid.

**Calling Sequence:** bc_points (bsploc, bgridp, bspaux, xcgrid, ycgrid, kgrpnt, xytst, kgrbnd, xp2, yp2, boun_coun, nbounc, donall)

**Data Declaration:**

| | |
|---|---|
| Real | xcgrid, ycgrid, xp2, yp2 |
| Integer | bsploc, bgridp, bspaux, kgrpnt, xytst, kgrbnd, boun_coun, nbounc |
| Logical | donall |

**Arguments:**

| | |
|---|---|
| bsploc | Place in array *bspecs* for storing interpolated spectra. |
| bgridp | Data concerning boundary grid points. |
| bspaux | Auxiliary array used for interpolation. |
| xcgrid | X-coordinate of computational grid points. |
| ycgrid | Y-coordinate of computational grid points. |
| kgrpnt | Indirect addresses of computational grid points. |
| xytst | Array of (*ix, iy*) of test points. |
| kgrbnd | Array of boundary grid points. |
| xp2 | Problem x-coordinate of a boundary location. |
| yp2 | Problem y-coordinate of a boundary location. |
| boun_coun | Counter show of the existing boundary point. |
| nbounc | Maximum number of boundary points. |
| donall | Declares if the nesting boundary is open or closed. *Donall* is defined by the users. |

#### 5.3.17.3 Subroutine BCWAMN

Subroutine BCWAMN reads file data for WAM nesting boundary conditions.

**Calling Sequence:** bcwamn (fbcnam, bctype, bfiled, bsploc, bspdir, rbsdir, bspfrq, rbsfrq, bgridp, bspaux, rbsaux, xcgrid, ycgrid, kgrpnt, xytst)

**Data Declaration:**

| | |
|---|---|
| Real | rbsaux, rbsdir, rbsfrq, xcgrid, ycgrid |
| Integer | bspaux, bspdir, bspfrq, bfiled, bgridp, kgrpnt, bsploc, xytst |
| Character | fbcnam, bctype |

| | | |
|---|---|---|
| **Arguments:** | fbcnam | Filename of boundary data file. |
| | bctype | If value is "NEST" → nesting b.c. |
| | bfiled | Data concerning boundary condition files. |
| | bsploc | Place in array *bspecs* that stores interpolated spectra. |
| | bspdir | Spectral directions of input spectrum. |
| | rbsdir | Real equivalence of *bspdir.* |
| | bspfrq | Spectral frequencies of input spectrum. |
| | rbsfrq | Real equivalence of *bspfrq.* |
| | bgridp | Data concerning boundary grid points. |
| | bspaux | Auxiliary array used for interpolation. |
| | rbsaux | Real equivalence of *bspaux.* |
| | xcgrid | X-coordinate of computational grid points. |
| | ycgrid | Y-coordinate of computational grid points. |
| | kgrpnt | Indirect addresses of grid points. |
| | xytst | *Ix, iy* of test points. |

#### 5.3.17.4 Subroutine BCWW3N

Subroutine BCWW3N reads file data for WAVEWATCH III boundary conditions.

**Calling Sequence:** bcww3n (fbcnam, bctype, bfiled, bsploc, bspdir, rbsdir, bspfrq, rbsfrq, bgridp, bspaux, xcgrid, ycgrid, kgrpnt, xytst, kgrbnd, donall)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xcgrid, ycgrid, rbsdir, rbsfrq |
| | Integer | bfiled, kgrbnd, xytst, kgrpnt, bgridp, bsploc, bspaux, bspdir, bspfrq |
| | Character | fbcnam, bctype |
| | Logical | donall |

| | | |
|---|---|---|
| **Arguments:** | fbcnam | Filename of boundary data file. |
| | bctype | Boundary condition type, is “WW3N” in this case. |
| | bfiled | Data concerning boundary condition files. |
| | bsploc | Place in array *bspecs* where to store interpolated spectra. |
| | bspdir | Spectral directions of input spectrum. |
| | rbsdir | Real equivalence of *bspdir.* |
| | bspfrq | Spectral frequencies of input spectrum. |
| | rbsfrq | Real equivalence of *bspfrq.* |
| | bgridp | Data concerning boundary grid points. |
| | bspaux | Auxiliary array used for interpolation. |
| | xcgrid | X-coordinate of computational grid points. |

| | |
|---|---|
| ycgrid | Y-coordinate of computational grid points. |
| kgrpnt | Indirect addresses of computational grid points. |
| xytst | Array of (*ix, iy*) of test points. |
| kgrbnd | Array of boundary grid points. |
| donall | Declares if the boundary is open or closed. |

### 5.3.17.5 Logical Function BOUNPT

Subroutine BOUNPT determines whether a grid point is a point where a boundary condition can be applied.

**Calling Sequence:** bounpt (ix, iy, kgrpnt)

**Data Declaration:** Integer ix, iy, kgrpnt

**Arguments:**

| | |
|---|---|
| ix, iy | Grid point indices. |
| kgrpnt | Indirect addresses of grid points. |

### 5.3.17.6 Subroutine RETSTP

Subroutine RETSTP reads test points, generates output point set TESTPNTS, and reads source term filenames.

**Calling Sequence:** retstp (mptst, xytst, kgrpnt, kgrbnd, xcgrid, ycgrid, spcsig, spcdir, ioutda, routda)

**Data Declaration:**

| | |
|---|---|
| Real | xcgrid, ycgrid, spcsig, spcdir, routda |
| Integer | mptst, xytst, kgrpnt, kgrbnd, ioutda |

**Arguments:**

| | |
|---|---|
| mptst | Maximum number of test points. |
| xytst | Grid point indices of test points. |
| kgrpnt | Indirect addresses of grid points. |
| kgrbnd | Array of boundary grid points. |
| xcgrid | X-coordinate of computational grid in x-direction. |
| ycgrid | Y-coordinate of computational grid in y-direction. |
| spcsig | Relative frequencies in the computational domain in sigma space. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |

| | |
|---|---|
| ioutd | Integer equivalence of *outda*. |
| routda | Real equivalence of *outda*. |

### 5.3.17.7 Function SIRAY

Subroutine SIRAY searches the first point on a ray where the depth is *dp*.

**Calling Sequence:** siray (dp, xp1, yp1, xp2, yp2, xx, yy, botdep, botlev, watlev)

**Data Declaration:**

| | |
|---|---|
| Logical | botdep |
| Real | botlev, watlev, xp1, yp1, xp2, yp2, xx, yy, dp |

**Arguments:**

| | |
|---|---|
| dp | Depth. |
| xp1 | X-coordinate start point of ray. |
| yp1 | Y-coordinate start point of ray. |
| xp2 | X-coordinate end point of ray. |
| yp2 | Y-coordinate end point of ray. |
| xx | X-coordinate point with depth *dp*. |
| yy | Y-coordinate point with depth *dp*. |
| botdep | Indicates that bottom depth is being read. |
| botlev | Bottom levels. |
| watlev | Water levels. |

### 5.3.17.8 Subroutine SPROUT

Subroutine SPROUT reads and processes the user output commands.

**Calling Sequence:** sprout (found, outda, routda, spcsig, xcgrid, ycgrid, kgrpnt, botlev, watlev)

**Data Declaration:**

| | |
|---|---|
| Real | routda, spcsig, xcgrid, ycgrid, botlev, watlev |
| Integer | outda, kgrpnt |
| Logical | found |

**Arguments:**

| | |
|---|---|
| found | Parameter indicating whether command being processed is found (value True) or not (False). |
| outda | Array containing output data. |
| routda | Real equivalence of *outda*. |
| spcsig | Relative frequencies in computational domain in sigma space. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |
| kgrpnt | Indirect addresses of the computational grid points. |

| | |
|---|---|
| botlev | Bottom levels. |
| watlev | Water levels. |

### 5.3.17.9 Subroutine SVARTP

Subroutine SVARTP converts keywords into an integer.

**Calling Sequence:** svartp (ivtype)

**Data Declaration:** Integer ivtype

**Arguments:** ivtype Type number output variable.

### 5.3.17.10 Subroutine SWBOUN

Subroutine SWBOUN reads and processes boundary commands.

**Calling Sequence:** swboun (bfiles, bsploc, rbsloc, bspdir, rbsdir, bspfrq, rbsfrq, bspecs, mxspec, bgridp, bspaux, rbsaux, xcgrid, ycgrid, kgrpnt, spcsig, spcdir, bcaux, xytst, kgrbnd)

**Data Declaration:**

| | |
|---|---|
| Real | rbsloc, rbsdir, rbsfrq, rbsaux, xcgrid, ycgrid, spcsig, spcdir, bspecs |
| Integer | bsploc, bspdir, bspfrq, bspaux, bfiles, mxspec, bcaux, bgridp, kgrpnt, xytst, kgrbnd |

**Arguments:**

| | |
|---|---|
| bfiles | Data concerning boundary condition files. |
| bsploc | Place in array *bspecs* where to store interpolated spectra. |
| rbsloc | Real equivalence of *bsploc.* |
| bspdir | Integer equivalence of *rbsdir.* |
| rbsdir | Spectral directions of input spectrum. |
| bspfrq | Integer equivalence of *rbsfrq.* |
| rbsfrq | Spectral frequencies of input spectrum. |
| bspecs | Array containing boundary spectra. |
| mxspec | Number of spectra that *bspecs* can contain. |
| bgridp | Data concerning boundary grid points. |
| bspaux | Auxiliary array used for interpolation. |
| rbsaux | Real equivalence of *bspaux.* |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |
| kgrpnt | Indirect addresses of grid points. |
| spcsig | Relative frequencies in the computational domain in |

| | |
|---|---|
| | sigma space. |
| spcdir | (*,1) Spectral directions (radians); |
| | (*,2) Cosine of spectral directions; |
| | (*,3) Sine of spectral directions; |
| | (*,4) Cosine^2 of spectral directions; |
| | (*,5) Cosine*sine of spectral directions; |
| | (*,6) Sine^2 of spectral directions. |
| bcaux | Auxiliary array used in this subroutine. |
| xytst | *Ix, iy* of test points. |
| kgrbnd | Array of boundary grid points. |

#### 5.3.17.11 Subroutine SWNMPS

Subroutine SWNMPS reads the name of the set of output points and gets the type and number of points in the set.

**Calling Sequence:** swnmps (outps, psname, pstype, mip, ierr)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | outps, mip, ierr |
| | Character | psname, pstype |
| **Arguments:** | outps | Array containing data on output point sets. |
| | psname | Output name. |
| | pstype | Output type. |
| | mip | Number of points. |
| | ierr | Error status:<br>= 0 No error;<br>= 9 End-of-file. |

#### 5.3.17.12 Subroutine SWREOQ

Subroutine SWREOQ reads and processes the output requests.

**Calling Sequence:** swreoq (found, outoq, outor, outps, outpr, spcsig)

| | | |
|---|---|---|
| **Data Declaration:** | Real | spcsig, outor, outpr |
| | Integer | outoq, outps |
| | Logical | found |
| **Arguments:** | found | Parameter indicating whether the command being processed is found (value True) or not (False). |
| | outoq | Array containing various parameters related to output requests (plotting). |

| | |
|---|---|
| outor | Array containing various parameters related to output requests (plotting). |
| outps | Array containing data on output point sets. |
| outpr | Real equivalence of *outps*. |
| spcsig | Relative frequencies in the computational domain in sigma space. |

#### 5.3.17.13 Subroutine SWREPS

Subroutine SWREPS reads and processes the commands defining output points.

**Calling Sequence:** swreps (found, outps, outpr, xcgrid, ycgrid, botlev, watlev)

**Data Declaration:**

| | |
|---|---|
| Real | xcgrid, ycgrid, botlev, watlev, outpr |
| Integer | outps |
| Logical | found |

**Arguments:**

| | |
|---|---|
| found | Parameter indicating whether command being processed is found (value True) or not (False). |
| outps | Array containing data on output point sets. |
| outpr | Real equivalence of *outps*. |
| xcgrid | X-coordinate of computational grid in x direction. |
| ycgrid | Y-coordinate of computational grid in y direction. |
| botlev | Bottom levels. |
| watlev | Water levels. |

### *5.3.18 SWAN Service Routines (swanser FOR File)*

#### 5.3.18.1 Subroutine AC2TST

**Calling Sequence:** ac2tst (xytst, ac2, kgrpnt)

**Data Declaration:**

| | |
|---|---|
| Integer | xytst, kgrpnt |
| Real | ac2 |

**Arguments:**

| | |
|---|---|
| xytst | Grid point indices of test points. |
| ac2 | Action density array. |
| kgrpnt | Array of indirect addressing. |

### 5.3.18.2 Real Function ANGDEG

Function ANGDEG transforms radians to degrees.

**Calling Sequence:** angdeg (radian)

| | | |
|---|---|---|
| **Data Declaration:** | Real | radian |
| **Arguments:** | radian | Radians. |

### 5.3.18.3 Real Function ANGRAD

Function ANGRAD transforms degrees to radians.

**Calling Sequence:** angrad (degree)

| | | |
|---|---|---|
| **Data Declaration:** | Real | degree |
| **Arguments:** | degree | Degrees. |

### 5.3.18.4 Subroutine CHGBAS

Subroutine CHGBAS changes the x-basis of a discretized y-function.

**Calling Sequence:** chgbas (x1, x2, period, y1, y2, n1, n2, itest, prtest)

| | | |
|---|---|---|
| **Data Declaration:** | Real | x1, x2, y1, y2, period |
| | Integer | n1, n2, itest, prtest |
| **Arguments:** | x1 | X-coordinate of input grid. |
| | x2 | X-coordinate of output grid. |
| | period | Period, i.e. x-axis is periodic if *period* > 0 e.g. spectral directions. |
| | y1 | Function values of input grid. |
| | y2 | Function values of output grid. |
| | n1 | Number of x-values of input grid. |
| | n2 | Number of x-values of output grid. |
| | itest | Integer variable which determines the level of test output. |
| | prtest | Unit number for output. |

### 5.3.18.5 Subroutine CVCHEK

Subroutine CVCHEK checks whether or not the given curvilinear grid is correct. CVCHEK also sets the value of *cvleft*.

| | | |
|---|---|---|
| **Calling Sequence:** | cvchek (kgrpnt, xcgrid, ycgrid) | |
| **Data Declaration:** | Integer | kgrpnt |
| | Real | xcgrid, ycgrid |
| **Arguments:** | kgrpnt | Array of indirect addressing. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |

### 5.3.18.6 Subroutine CVMESH

Subroutine CVMESH finds location in a curvilinear grid for a point given in problem coordinates.

| | | |
|---|---|---|
| **Calling Sequence:** | cvmesh (xp, yp, xc, yc, kgrpnt, xcgrid, ycgrid, kgrbnd) | |
| **Data Declaration:** | Real | xcgrid, ycgrid, xp, yp, xc, yc |
| | Integer | kgrpnt, kgrbnd |
| **Arguments:** | xp, yp | A point given in problem coordinates. |
| | xc, yc | Same point in computational grid coordinates. |
| | kgrpnt | Array (*mxc, myc*) grid numbers if *kgrpnt* <= 1, the point is not in computational grid. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |
| | kgrbnd | Lists all boundary grid points consecutively. |

### 5.3.18.7 Real Function DEGCNV

Function DEGCNV transforms degrees from Nautical to Cartesian or vice versa.

| | | |
|---|---|---|
| **Calling Sequence:** | degcnv (degree) | |
| **Data Declaration:** | Real | degree |
| **Arguments:** | degree | Direction in Nautical or Cartesian degrees. |

### 5.3.18.9 Subroutine EVALF

Subroutine EVALF evaluates the coordinates (in problem coordinates) of point (*xc, yc*) given in computational coordinates.

| | | |
|---|---|---|
| **Calling Sequence:** | evalf (xc, yc, xvc, yvc, xcgrid, ycgrid) | |
| **Data Declaration:** | Real | xc, yc, xvc, yvb, xcgrid, ycgrid |
| **Arguments:** | xc, yc | Point in computational grid coordinates. |
| | xvc, yvc | Same point but in problem coordinates. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |

### 5.3.18.10 Real Function GAMMA

Function GAMMA computes the transcendental function GAMMA.

| | | |
|---|---|---|
| **Calling Sequence:** | gamma (xx) | |
| **Data Declaration:** | Real | xx |
| **Arguments:** | xx | X-coordinate of the point. |

### 5.3.18.11 Function GAMMLN

| | | |
|---|---|---|
| **Calling Sequence:** | gammln (xx) | |
| **Data Declaration:** | Real | xx |
| **Arguments:** | xx | X-coordinate of the point. |

### 5.3.18.12 Subroutine HSOBND

Subroutine HSOBND compares computed significant wave height with the value of the significant wave height as described by the user. If the values differ more than, say, ten percent, an error message and the grid points where the error has been located are given.

| | | |
|---|---|---|
| **Calling Sequence:** | hsobnd (ac2, spcsig, hsibc, kgrpnt) | |
| **Data Declaration:** | Real | ac2, spcsig, hsibc |
| | Integer | kgrpnt |

| | | |
|---|---|---|
| **Arguments:** | ac2 | Action density. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | hsibc | Significant wave height given as input on the boundary. |
| | kgrpnt | Values of grid indices. |

#### 5.3.18.13 Logical Function INFRAM

Subroutine INFRAM checks whether a point given in frame coordinates is located in the plotting frame (INFRAM = True) or not (INFRAM = False).

**Calling Sequence:** infram (xqq, yqq)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xqq, yqq |

| | | |
|---|---|---|
| **Arguments:** | xqq | X-coordinate (output grid) of the point. |
| | yqq | Y-coordinate (output grid) of the point. |

#### 5.3.18.14 Logical Function INMESH

Function INMESH finds whether or not a given location is in the (curvilinear) computational grid.

**Calling Sequence:** inmesh (xp, yp, xcgrid, ycgrid, kgrbnd)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xp, yp, xcgrid, ycgrid |
| | Integer | kgrbnd |

| | | |
|---|---|---|
| **Arguments:** | xp, yp | A point given in problem coordinates. |
| | xcgrid | Array (*ix, iy*) x-coordinate of a grid point. |
| | ycgrid | Array (*ix, iy*) y-coordinate of a grid point. |
| | kgrbnd | Array containing boundary grid points. |

#### 5.3.18.15 Subroutine KSCIP1

Subroutine KSCIP1 interpolates the wave number, group velocity and *n* from a table, and calculation of the derivative of *n* with respect to depth (= *nd*).

| | | |
|---|---|---|
| **Calling Sequence:** | kscip1 (mmt, sig, d, k, cg, n, nd) | |
| **Data Declaration:** | Integer | mmt |
| | Real | sig, d, k, cg, n, nd |
| **Arguments:** | mmt | Number of frequency-wise points in arrays. |
| | sig | Relative frequency for which wave parameters must be determined. |
| | d | Local depth. |
| | k | Wave number. |
| | cg | Group velocity. |
| | n | Ratio of group and phase velocity. |
| | nd | Derivative of *n* with respect to *d* computation must be done. |

#### 5.3.18.16 Subroutine NEWTON

Subroutine NEWTON solves equations and finds a point (*xc, yc*) in a curvilinear grid (computational grid) for a given point (*xp, yp*) in a Cartesian grid (problem coordinates).

| | | |
|---|---|---|
| **Calling Sequence:** | newton (xp, yp, xcgrid, ycgrid, kgrpnt, mxitnr, xc, yc, find, kgrbnd) | |
| **Data Declaration:** | Real | xp, yp, xcgrid, ycgrid, xc, yc |
| | Integer | kgrbnd |
| | Logical | find |
| **Arguments:** | xp | X-coordinate in problem coordinates. |
| | yp | Y-coordinate in problem coordinates. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |
| | kgrpnt | Grid addresses. |
| | mxitnr | Maximum number of iterations. |
| | xc | X-coordinate in computational coordinates. |
| | yc | Y-coordinate in computational coordinates. |
| | find | Determines whether or not *xc* and *yc* are found. |
| | kgrbnd | Grid addresses of the boundary points. |

#### 5.3.18.17 Subroutine NEWT1D

Subroutine NEWT1D solves equations and finds a point (*xc, yc*) in a curvilinear 1-D grid (computational grid) for a given point (*xp, yp*) in a Cartesian grid (problem coordinates).

**Calling Sequence:** newt1d (xp, yp, xcgrid, ycgrid, kgrpnt, mxitnr, xc, yc, find)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xp, yp, scgrid, ycgrid, xc, yc |
| | Integer | kgrpnt, mxitnr |
| | Logical | find |
| **Arguments:** | xp | X-coordinate in problem coordinates. |
| | yp | Y-coordinate in problem coordinates. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |
| | kgrpnt | Grid addresses. |
| | mxitnr | Maximum number of iterations. |
| | xc | X-coordinate in computational coordinates. |
| | yc | Y-coordinate in computational coordinates. |
| | find | Whether or not *xc* and *yc* are found. |

### 5.3.18.18 Subroutine OBSTLINE

Subroutine OBSTLINE finds out whether or not vector (*x1, y1*) lies above the line piece through (*x3, y3*) and (*x4, y4*).

**Calling Sequence:** obstline (x1, y1, x2, y2, x3, y3, x4, y4, xgtl, exc)

| | | |
|---|---|---|
| **Data Declaration:** | Real | x1, y1, x2, y2, x3, y3, x4, y4 |
| | Logical | xgtl, exc |
| **Arguments:** | x1, y1 | User coordinates of one end of the grid link. |
| | x2, y2 | User coordinates of the other end of the grid link. |
| | x3, y3 | User coordinates of one end of the obstacle side. |
| | x4, y4 | User coordinates of the other end of the obstacle side. |
| | xgtl | Indicates whether (*x1, y1*) is situated above line piece (*x3, y3*) (*x4, y4*). |
| | exc | Indicates whether *x4* = *x3*, which results in exceptional situation (line parallel to y-axis). |

### 5.3.18.19 Recursive Subroutine OBSTMOVE

Subroutine OBSTMOVE moves obstacle points (*x3, y3*) and (*x4, y4*) a bit if computational grid cell (*x1, y1*) is on the obstacle line piece.

**Calling Sequence:** obstmove (obsta, xcgrid, ycgrid, kgrpnt)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xcgrid, ycgrid |

| | | |
|---|---|---|
| | Integer | obsta, kgrpnt |
| **Arguments:** | obsta | Array of obstacle parameters. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |
| | kgrpnt | Indirect addressing for computational grid points. |

#### 5.3.18.20 Subroutine PCOAST

Subroutine PCOAST plots lines defined by the command LINE.

**Calling Sequence:** pcoast (clines, cliner)

| | | |
|---|---|---|
| **Data Declaration:** | Real | cliner |
| | Integer | clines |
| **Arguments:** | clines | Line parameter. |
| | cliner | Real equivalence to *clines*. |

#### 5.3.18.21 Subroutine PLNAME

Subroutine PLNAME writes the name of a place or region in a plot.

**Calling Sequence:** plname (pname, nsym, xpp, ypp, isit, symsz)

| | | |
|---|---|---|
| **Data Declaration:** | Character | pname |
| | Real | xpp, ypp, symsz |
| | Integer | nsym, isit |
| **Arguments:** | pname | Name of town or region to be plotted. |
| | nsym | Number of characters of the name. |
| | xpp | X-coordinate of the reference point in the problem grid. |
| | ypp | Y-coordinate of the reference point in the problem grid. |
| | isit | Type of name (0 or 1: the name is plotted right of the reference point with (1) or without (0) a mark at the point, 2: the reference point is at the middle of the name (region)). |
| | symsz | Size of the characters in the plot (cm). |

### 5.3.18.22 Subroutine PLOSIT

Subroutine PLOSIT draws a plot with the location of the output point sets.

**Calling Sequence:** plosit (outps, outpr, psname)

**Data Declaration:**

| | |
|---|---|
| Character | psname |
| Real | outpr |
| Integer | outps |

**Arguments:**

| | |
|---|---|
| outps | Array containing data on output point sets. |
| outpr | Real equivalence of *outps*. |
| psname | Name of one output point set to be plotted if blank, all point sets will be plotted. |

### 5.3.18.23 Subroutine PLOTU

Subroutine PLOTU moves the pen to a point given in problem coordinates with pen up (moving the pen) or with pen down (drawing a line segment).

**Calling Sequence:** plotu (xx, yy, updown)

**Data Declaration:**

| | |
|---|---|
| Real | xx, yy |
| Character | updown |

**Arguments:**

| | |
|---|---|
| xx | X-coordinate of the point. |
| yy | Y-coordinate of the point. |
| updown | Indicating whether the pen must be up or down when moving to the point. |

### 5.3.18.24 Subroutine PNAMES

Subroutine PNAMES plots the names of places and regions defined with the command PLACE.

**Calling Sequence:** pnames (places, placer)

**Data Declaration:**

| | |
|---|---|
| Integer | places |
| Real | placer |

**Arguments:**

| | |
|---|---|
| places | Array containing places and their locations. |
| placer | Real equivalence of *places*. |

### 5.3.18.25 Subroutine READXY

Subroutine READXY reads x and y and initializes offset values xoffs and yoffs.

**Calling Sequence:** readxy (namx, namy, xx, yy, kont, xsta, ysta)

**Data Declaration:**

| | |
|---|---|
| Real | xx, yy, xsta, ysta |
| Character | namx, namy, kont |

**Arguments:**

| | |
|---|---|
| namx, namy | Names of the two coordinates given in the user manual. |
| xx, yy | Values of x and y, taking into account offset. |
| kont | If values are missing see documentation of INDBLE (Ocean Pack documentation). |
| xsta, ysta | Standard values of x and y. |

### 5.3.18.26 Subroutine REFIXY

Subroutine REFIXY initializes offset values xoffs and yoffs, and shifts *xx* and *yy*.

**Calling Sequence:** refixy (nds, xx, yy, ierr)

**Data Declaration:**

| | |
|---|---|
| Real | xx, yy |
| Integer | nds, ierr |

**Arguments:**

| | |
|---|---|
| nds | File reference number. |
| xx, yy | Values of x and y taking into account offset. |
| ierr | Error indicator: When *ierr*:<br>= 0 No error;<br>= -1 End-of-file;<br>= -2 Read error. |

### 5.3.18.27 Subroutine REFLECT

Subroutine REFLECT computes reflections near obstacles.

**Calling Sequence:** reflect (ac2, ac2ref, imatra, x1, y1, x2, y2, x3, y3, x4, y4, xgtl, exc, cax, cay, rdx, rdy, loop, trcoef, ref0, anybin)

**Data Declaration:**

| | |
|---|---|
| Integer | loop |
| Real | ac2, imatra, x1, y1, x2, y2, x3, y3, x4, y4, cax, cay, rdx, rdy, ac2ref, trcoef, ref0 |
| Logical | anybin, xgtl, exc |

**Arguments:**

| | | |
|---|---|---|
| | ac2 | (Non-stationary case) action density as function of D, S, X, Y at time T + DT. |
| | ac2ref | (Non-stationary case) reflected action density as function of D, S, X, Y at time T + DT. |
| | imatra | Right-hand side of matrix equation. |
| | x1, y1 | Coordinates of computational grid point under consideration. |
| | x2, y2 | Coordinates of computational grid point neighbor. |
| | x3, y3 | User coordinates of one end of obstacle side. |
| | x4, y4 | User coordinates of the other end of the obstacle side. |
| | xgtl | Indicates whether (*x1, y1*) is situated above line piece (*x3, y3*) (*x4, y4*). |
| | exc | Indicates whether *x4* = *x3*, which results in an exception situation (line parallel to y-axis). |
| | cax, cay | Propagation velocity. |
| | rdx, rdy | Array containing spatial derivative coefficients. |
| | loop | Indicates which link is analyzed:<br>1 $\rightarrow$ neighbor in x;<br>2 $\rightarrow$ neighbor in y. |
| | trcoef | User defined transmission coefficient. |
| | ref0 | User defined reflection coefficient (0 <= *ref0* <= 1). |
| | anybin | Set a particular bin True or False depending on *sector*. |

### 5.3.18.28 Subroutine SETUPP

Subroutine SETUPP computes the forces/(*rho***grav*) responsible for the *setup* and adds the *setup* to the depth.

**Calling Sequence:** setupp (kgrpnt, mstpda, setpda, ac2, dep2, depsav, setup2, wforcx, wforcy, xcgrid, ycgrid, spcsig, spcdir, itsw, iter, upperi, loperi)

**Data Declaration:**

| | | |
|---|---|---|
| | Real | setpda, ac2, dep2, depsav, setup2, wforcx, wforcy, xcgrid, ycgrid, spcsig, spcdir, upperi, loperi |
| | Integer | kgrpnt, mstpda, itsw, iter |

**Arguments:**

| | | |
|---|---|---|
| | kgrpnt | Indirect addresses for grid points. |
| | mstpda | Number of (aux.) data per grid point value is set at 10 in swancom1.ftn. |
| | setpda | Data for computation of Setup:<br>= 1 Depth; |

| | |
|---|---|
| | = 2 Previous estimate of Setup; |
| | = 3 X-comp of force; |
| | = 4 Y-comp of force; |
| | = 5 Rad. stress computation RSxx; |
| | = 6 RSxy; |
| | = 7 RSyy. |
| | *setpda*(*, *, 5 *mstpda*) is used as a work array. |
| ac2 | Action density as a function of D, S, X, Y at time T + DT. |
| dep2 | Total depth, including Setup on entry: includes previous estimate of Setup on exit: includes new estimate of Setup. |
| depsav | Depth following from bottom and water levels. |
| setup2 | Setup in grid points, using indirect addresses. |
| wforcx | Force x-component. |
| wforcy | Force y-component. |
| xcgrid | X-coordinate of computational grid in x-direction. |
| ycgrid | Y-coordinate of computational grid in y-direction. |
| spcsig | Relative frequencies in computational domain in sigma space. |
| spcdir | (*,1) Spectral directions (radians); |
| | (*,2) Cosine of spectral directions; |
| | (*,3) Sine of spectral directions; |
| | (*,4) Cosine^2 of spectral directions; |
| | (*,5) Cosine*sine of spectral directions; |
| | (*,6) Sine^2 of spectral directions. |
| itsw | Time step counter for SWAN. |
| iter | Iteration counter. |
| upperi | Only relevant for computation in periodic domain. |
| loperi | Only relevant for computation in periodic domain. |

#### 5.3.18.29 Subroutine SETUP2D

Subroutine SETUP2D computes the *setup*, change of the water level by waves. A Poisson equation is solved in general coordinates.

**Calling Sequence:** setup2d (xcgrid, ycgrid, wfrcx, wfrcy, depth, setup, upperi, loperi, nwkarr, wkarr, itsw, iter)

**Data Declaration:**

| | | |
|---|---|---|
| Integer | itsw, ier, nwkarr | |
| Real | xcgrid, ycgrid, wfrcx, wfrcy, depth, setup, upperi, loperi, nwkarr, wkarr, itsw, iter | |

**Arguments:**

| | |
|---|---|
| xcgrid | X-coordinates. |

| | |
|---|---|
| ycgrid | Y-coordinates. |
| wfrcx | Force x-component. |
| wfrcy | Force y-component. |
| depth | Depth. |
| setup | Unknown setup; to be computed indirect addressed. |
| upperi | Only relevant for computation in periodic domain. |
| loperi | Only relevant for computation in periodic domain. |
| nwkarr | Dimension for work array. |
| wkarr | Work array. |
| itsw | Time step counter for SWAN. |
| iter | Iteration number. |

### 5.3.18.30 Subroutine SINTRP

Subroutine SINTRP interpolates spectra.

**Calling Sequence:** sintrp (w1, w2, fl1, fl2, fl, spcdir, spcsig)

**Data Declaration:** Real w1, w2, fl1, fl2, fl, spcdir, spcsig

**Arguments:**

| | |
|---|---|
| w1 | Weighting coefficient for spectrum one. |
| w2 | Weighting coefficient for spectrum two. |
| fl1 | Input spectrum one. |
| fl2 | Input spectrum two. |
| fl | Interpolated spectrum. |
| spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| spcsig | Relative frequencies in computational domain in sigma space. |

### 5.3.18.31 Subroutine SSHAPE

Subroutine SSHAPE calculates energy density at boundary point (x, y, sigma, theta).

**Calling Sequence:** sshape (acloc, spcsig, spcdir, fshapl, dshapl)

**Data Declaration:**

| | |
|---|---|
| Real | acloc, spcsig, spcdir |
| Integer | fshapl, dshapl |

| | | |
|---|---|---|
| **Arguments:** | acloc | Energy density at a point in space. |
| | spcsig | Relative frequencies in computational domain in sigma space. |
| | spcdir | (*,1) Spectral directions (radians);<br>(*,2) Cosine of spectral directions;<br>(*,3) Sine of spectral directions;<br>(*,4) Cosine^2 of spectral directions;<br>(*,5) Cosine*sine of spectral directions;<br>(*,6) Sine^2 of spectral directions. |
| | fshapl | Shape of spectrum:<br>= 1 Pierson-Moskowitz spectrum;<br>= 2 JONSWAP spectrum;<br>= 3 bin;<br>= 4 Gauss curve;<br>If > 0 Period is interpreted as peak per;<br>If < 0 Period is interpreted as mean per. |
| | dshapl | Directional distribution. |

**Common Blocks:** PSHAPE
SPPARM

### 5.3.18.32 Subroutine SWOBST

Subroutine SWOBST reads from the *pool* array all the data required to find obstacles and uses subroutine TCROSS2 to find them.

**Calling Sequence:** swobst (obsta, xcgrid, ycgrid, kgrpnt, cross)

| | | |
|---|---|---|
| **Data Declaration:** | Real | xcgrid, ycgrid |
| | Integer | kgrpnt, obsta, cross |

| | | |
|---|---|---|
| **Arguments:** | obsta | Array of obstacle parameters. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |
| | kgrpnt | Indirect addressing for computational grid points. |
| | cross | Array that contains 0's if there is no obstacle crossing. If an obstacle is crossing between the central point and its neighbor, *cross* is equal to the number of the obstacles. |

### 5.3.18.33 Subroutine SWTRCF

Subroutine SWTRCF takes the value of transmission coefficient from the pool given by the user for obstacle transmission or computes the transmission coefficient for obstacle DAM, based on Goda (1967) [from Seelig (1979)]. If reflections are turned on, the source term in subroutine REFLECT is calculated.

**Calling Sequence:** swtrcf (obsta, cross, wlev2, chs, link, obredf, ac2, imatra, kgrpnt, xcgrid, ycgrid, cax, cay, rdx, rdy, anybin)

| | | |
|---|---|---|
| **Data Declaration:** | Integer | cross, obsta, kgrpnt, link |
| | Real | chs, obredf, wlev2, ac2, xcgrid, ycgrid, cax, cay, imatra, rdx, rdy |
| | Logical | anybin |

| | | |
|---|---|---|
| **Arguments:** | obsta | Array containing obstacle data. |
| | cross | Array that contains 0's if there is no obstacle crossing. If an obstacle is crossing between the central point and its neighbor; *cross* is equal to the number of the obstacle. |
| | wlev2 | Water level in grid points. |
| | chs | Hs in all computational grid points. |
| | link | Indicates whether link in stencil crosses an obstacle. |
| | obredf | Array of action density reduction coefficients (reduction at the obstacle). |
| | ac2 | Action density array. |
| | imatra | Coefficients of right-hand side of matrix equation. |
| | kgrpnt | Array of indirect addressing. |
| | xcgrid | X-coordinate of computational grid in x-direction. |
| | ycgrid | Y-coordinate of computational grid in y-direction. |
| | cax, cay | Propagation velocities. |
| | rdx, rdy | Array containing spatial derivative coefficients. |
| | anybin | Set a particular bin True or False depending on *sector*. |

### 5.3.18.34 Logical Function TCROSS

Function TCROSS finds out if there is an obstacle crossing the stencil being used.

**Calling Sequence:** tcross (x1, x2, x3, x4, y1, y2, y3, y4)

**Data Declaration:** Real x1, x2, x3, x4, y1, y2, y3, y4

| | | |
|---|---|---|
| **Arguments:** | x1, y1 | User coordinates of one end of grid link. |
| | x2, y2 | User coordinates of the other end of grid link. |
| | x3, y3 | User coordinates of one end of the obstacle side. |
| | x4, y4 | User coordinates of the other end of the obstacle side. |

#### 5.3.18.35 Logical Function TCROSS2

Function TCROSS2 finds out if there is an obstacle crossing the stencil being used.

**Calling Sequence:** tcross2 (x1, x2, x3, x4, y1, y2, y3, y4, x1onobst)

| | | |
|---|---|---|
| **Data Declaration:** | Real | x1, x2, x3, x4, y1, y2, y3, y4 |
| | Logical | x1onobst |

| | | |
|---|---|---|
| **Arguments:** | x1, y1 | User coordinates of one end of the grid link. |
| | x2, y2 | User coordinates of the other end of the grid link. |
| | x3, y3 | User coordinates of one end of the obstacle side. |
| | x4, y4 | User coordinates of the other end of the obstacle side. |
| | x1onobst | Boolean which tells whether (*x1, y1*) is on obstacle. |

#### 5.3.18.36 Subroutine WRSPEC

Subroutine WRSPEC writes the action density spectrum in SWAN standard format.

**Calling Sequence:** wrspec (nref, acloc)

| | | |
|---|---|---|
| **Data Declaration:** | Real | acloc |
| | Integer | nref |

| | | |
|---|---|---|
| **Arguments:** | nref | Unit reference number or output file. |
| | acloc | 2-D spectrum or source term at one output location. |

### *5.3.19 Module Containing Global Variables (swmod1 FOR File)*

This file is used to create global variables used in whitecapping and integral parameter subroutines. It contains no subroutines.

# 7.0 NOTES

## 7.1 ACRONYMS AND OTHER ABBREVIATIONS

| | |
|---|---|
| ASCE | American Society of Civil Engineering |
| ASCII | American Standard Code for Information Interchange |
| BI-CGSTAB | Method to solve an asymmetric system of linear equations |
| BLAS | Basic Linear Algebra Subprograms |
| BSBT | Backward Space, Backward Time |
| CFL criterion | Courant-Friedrich-Levy condition for computational stability |
| DIA | Discrete Interaction Approximation |
| DTA | Discrete Triad Approximation |
| DUT frame | Delft University of Technology |
| EOF | End of File |
| GSE | Garden-Sprinkler Effect |
| HISWA | HIndcast Shallow Water wave model |
| ID | Identification |
| IUTAM | International Union of Theoretical and Applied Mechanics |
| JONSWAP | JOint North Sea WAve Project |
| LTA | Lumped Triad Approximation |
| Mb | Megabytes |
| OPPL | Ocean Pack PLot code |
| QB | Fraction of breaking waves |
| S&L | Stelling and Leendertse's second order with third-order diffusion scheme |
| SIAM | Society for Industrial and Applied Mathematics |
| SORDUP | Second ORDer Upwind scheme |
| SWAN | Simulating WAves Nearshore |
| WAM | WAve Model |
| WAMDI | WAM Development and Implementation group |

# TABLE OF COMMON BLOCKS

# 8.0 APPENDIX I.

## 8.1 (OCPIDS FOR FILE)

### *8.1.1 COMMON/ FILENM*
Filename of plot file.

### *8.1.2 COMMON/ XASL*
Size on paper of geographic area in x-direction.

### *8.1.3 COMMON/ YASL*
Size on paper of geographic area in y-direction.

### *8.1.4 COMMON/ SYMSIZ*
Size of symbols on plot.

### *8.1.5 COMMON/ XPLO*
Lowest x on paper of geographic area.

### *8.1.6 COMMON/ XPHI*
Highest x on paper of geographic area.

### *8.1.7 COMMON/ YPLO*
Lowest y on paper of geographic area.

### *8.1.8 COMMON/ YPHI*
Highest y on paper of geographic area.

### *8.1.9 COMMON/ SUBLNS*
Number of lines in caption for scales etc.

### *8.1.10 COMMON/ XPSUB*
Position of one line of caption.

### *8.1.11 COMMON/ YPSUB*
Y position of one line of caption.

### *8.1.12 COMMON/ PLPARM(3)*
Conversion factor; default 402.

### *8.1.13 COMMON/ PLPARM(4)*
Plotting margin horizontal.

#### *8.1.14 COMMON/ PLPARM(5)*
Plotting margin vertical.

#### *8.1.15 COMMON/ PLPARM(6)*
Rotation.

## 8.2 (OCPLOT FOR FILE)

#### *8.2.1 COMMON/ PMR*
Plot margin.

#### *8.2.2 COMMON/ MXQ*
Number of grid points in x-direction.

#### *8.2.3 COMMON/ MYQ*
Number of grid points in y-direction.

#### *8.2.4 COMMON/ DXQ*
Mesh size in x-direction.

#### *8.2.5 COMMON/ DYQ*
Mesh size in y-direction.

## 8.3 (OCPMIX FOR FILE)

#### *8.3.1 COMMON/ REFDAY*
Day number of the reference day; the reference time is 0:00 of the reference day; the first day entered is used as reference day.

## 8.4 (SWANMAIN FOR FILE)

#### *8.4.1 COMMON/ NAMES*
Names and other character strings.

| **Variable** | **Type** | **Description** |
|---|---|---|
| INST | Character | Name of the institute. It can be changed in the file SWANINIT. |
| PROJID | Character | Acronym of the project for which the computation is taking place. |
| PROJNR | Character | = BLANK; run number for the computation;<br>= NR; set by command PROJ ... NR ... |

| | | |
|---|---|---|
| PROJT1 | Character | = BLANK; first line of the project title;<br>= title1; set by command PROJ ... title1. |
| PROJT2 | Character | = BLANK; second line of the project title;<br>= title2; set by command PROJ ... title2. |
| PROJT3 | Character | = BLANK; third line of the project title;<br>= title3; set by command PROJ ... title3. |
| PTITLE | Character | Not used. |
| FILENM | Character | Filename of the file currently used for I/O. |
| FILEA | Character | Not used. |
| FILEB | Character | Not used. |
| DIRCH1 | Character | Directory separation character as appears in input file. |
| DIRCH2 | Character | Directory separation character replacing DIRCH1. |
| VERTXT | Character | Program version, character representation. |
| C4(LNAMS) | Character | Contains all the items in /NAMES/. C4 is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.2 COMMON/ TESTDA*

Test parameter.

| Variable | Type | Description |
|---|---|---|
| ITEST | Integer | Indicates the amount of test output requested. |
| ITRACE | Integer | Message is printed up to ITRACE times. |
| LTRACE | Logical | Indicates whether to call STRACE. |
| LEVERR | Integer | Severity of the errors encountered. |
| MAXERR | Integer | Maximum severity of errors allowed, if larger no computation:<br>= 1 Warnings;<br>= 2 Errors;<br>= 3 Severe errors;<br>= 4 Terminating errors;<br>= MAXERR Set by command SET ... [MAXERR]. |
| OTSTD(NTSTD) | Real | (Not used); Contains all of the items in /TESTDA/. OTSTD is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.3 COMMON/ OUTPDA*

Data for output, mainly plotting.

| Variable | Type | Description |
|---|---|---|
| LEFT | Logical | The coordinate system is left/right-oriented i.e. counterclockwise from X to Y/clockwise from Y to X. |
| PFROPT | Integer | Frame option in plot, read from SWANINIT. |
| VERNUM | Real | Version number of SWAN. |

| | | |
|---|---|---|
| XASM | Real | Maximum size of area available for plotting isolines and vector fields in x-direction. |
| YASM | Real | Maximum size of area available for plotting isolines and vector fields in y-direction. |
| MXQ | Integer | Number of grid points of the output frame in X-direction. |
| MYQ | Integer | Number of grid points of the output frame in Y-direction. |
| DXQ | Real | Mesh size of the output frame in X-direction. |
| DYQ | Real | Mesh size of the output frame in Y-direction. |
| XASL | Real | Size on paper of geographic area in x-direction. |
| YASL | Real | Size on paper of geographic area in y-direction. |
| SYMSIZ | Real | Size of the symbols in the plot. |
| LSC | Real | Not used. |
| VSC | Real | Not used. |
| PENUP | Logical | Not used. |
| XPLO | Real | Lowest x on paper of geographic area. |
| XPHI | Real | Highest x on paper of geographic area. |
| YPLO | Real | Lowest y on paper of geographic area. |
| YPHI | Real | Highest y on paper of geographic area. |
| HORSC | Real | Horizontal scale. |
| VRTSC | Real | Vertical scale. |
| XFLO | Real | Lower limit of X in the physical plane. |
| XFHI | Real | Upper limit of X in the physical plane. |
| YFLO | Real | Lower limit of Y in the physical plane. |
| YFHI | Real | Upper limit of Y in the physical plane. |
| SUBLNS | Integer | Number of lines available in the plot legend<br>= 3 If FROPT = 1<br>= 4 If FROPT = 2 |
| XPSUB | Real | Place (X-coordinate) of the legends in the frame. |
| YPSUB | Real | Place (Y-coordinate) of the legends in the frame. |
| ODA(MCODA) | Real | (Not used); Contains all the items in /OUTPDA/. ODA is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.4 COMMON/ REFNRS*

File unit reference numbers.

| **Variable** | **Type** | **Description** |
|---|---|---|
| PRINTF | Integer | Unit number for the file with standard output (PRINT). |
| INPUTF | Integer | Unit number for the file with command input (INPUT). |
| IUNMIN | Integer | Minimum unit number. |
| IUNMAX | Integer | Maximum unit number. |
| FUNLO | Integer | Lowest free unit number. |

| | | |
|---|---|---|
| FUNHI | Integer | Highest free unit number. |
| SCREEN | Integer | Unit number for the screen. |
| PRTEST | Integer | Unit number for the print file containing test output. |
| IMPORT | Integer | Not used. |
| EXPORT | Integer | Not used. |
| HIOPEN | Integer | Highest unit number of an open file. |
| ITMOPT | Integer | Time coding option. |
| IRFNS(NRFNS) | Integer | (Not used); Contains all of the items in /REFNRS/. IRFNS is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.5 COMMON/ LEESDA*

Character data used by the command reading system.

| Variable | Type | Description |
|---|---|---|
| ELTYPE | Character | Type of the element last read by reading system. |
| ELTEXT | Character | Contents of the last string read by reading system. |
| KAART | Character | Contents of the input line last read by the reading system. |
| KAR | Character | Character last read by the reading system. |
| KEYWRD | Character | Contents of the last keyword read by reading system. |
| BLANK | Character | Blank string. |
| TABC | Character | Tabular character. |
| COMID | Character | Character that distinguishes comments in the command input. |
| LSDA(NLSDA) | Character | Contains all of the items in /LEESDA/. LSDA is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.6 COMMON/ LEESDN*

Number data used by the command reading system.

| Variable | Type | Description |
|---|---|---|
| ELREAL | Double Precision | Last element read from user command, when real or double. |
| ELLINT | Integer | Last element read from user command, when integer. |
| KARNR | Integer | Position on the input line of character last processed by the reading system:<br>= 0 No characters read yet;<br>= 81 Next input line has to be read to the common KAART first. |
| CHGVAL | Logical | Whether or not the last read value is different from a given value for subroutines INREAL, ININTG, INCSTR and INCTIM. |
| LENCST | Integer | Length of the string stored in ELTEXT. |

| | | |
|---|---|---|
| ILSDN(NLSDN) | Integer | (Not used); Contains all of the items in /LEESDN/. ILSDN is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.7 COMMON/ SWNAME*

Names and other character data.

| Variable | Type | Description |
|---|---|---|
| FNEST<br>SNAME | Character<br>Character | Name of nest file.<br>Name of output point set. |
| OVKEYW | Character | Keyword identifying output quantity in a SWAN command. |
| OVSNAM | Character | Short name of output quantity. |
| OVLNAM | Character | Long name of output quantity. |
| OVUNIT | Character | Unit of output quantity. |
| UH | Character | Unit of vertical length (m). |
| UV | Character | Unit of velocity (m/s). |
| UT | Character | Unit of time (sec). |
| UL | Character | Unit of horizontal length (m). |
| UET | Character | Unit of energy transport, and wave force ($m^3/s$). |
| UDI | Character | Unit of direction (degrees). |
| UST | Character | Not used. |
| UF | Character | Unit of pressure or shear stress (force per area) ($N/m^2$). |
| UP | Character | Unit of energy flux density (W/m). |
| UAP | Character | Unit of dissipation ($W/m^2$). |
| UDL | Character | Unit of dissipation ($m^2/s$). |
| UD | Character | Not used. |
| FBCL | Character | Not used. |
| FBCR | Character | Not used. |
| CHTIME | Character | Character string representation of date-time of computation. |
| TIT(LHNAMS) | Character | Contains all of the items in /SWNAME/. TIT is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.8 COMMON/ SWGRID*

Location and dimensions of input grids.

| Variable | Type | Description |
|---|---|---|
| XPG | Real | X of origin. |
| YPG | Real | Y of origin. |
| ALPG | Real | Direction of the x-axis with respect to the user coordinates. |
| COSPG | Real | Cosine of ALPG. |

| | | |
|---|---|---|
| SINPG | Real | Sine of ALPG. |
| DXG | Real | Mesh size of input grid in x-direction. |
| DYG | Real | Mesh size of input grid in y-direction. |
| MXG | Integer | Number of meshes in x-direction. |
| MYG | Integer | Number of meshes in y-direction. |
| LEDS | Integer | = 0 When values have not been read;<br>= 1 If values were read. |
| IGTYPE | Integer | = 0 When grid has constant values;<br>= 1 When grid is regular;<br>= 2 When grid is curvilinear. |
| VARFR | Logical | Friction coefficient is or is not variable over space. |
| VARWI | Logical | Wind velocity is or is not variable over space. |
| COSVC | Real | Cosine of the angle of current input grid with respect to the computational grid. |
| SINVC | Real | Sine of the angle of current input grid with respect to the computational grid. |
| COSWC | Real | Cosine of the angle of wind input grid with respect to the computational grid. |
| SINWC | Real | Sine of the angle of wind input grid with respect to the computational grid. |
| XOFFS | Real | Offset value in x. |
| YOFFS | Real | Offset value in y. |
| LXOFFS | Logical | Offset values were or were not initialized already. |
| VARWLV | Logical | Water level is or is not variable over space. |
| DYNDEP | Logical | True if depth varies with time. |
| NESRUN | Integer | Indicator for a nested run. |
| NWAMN | Integer | Indicator for a WAM-nested run. |
| OPTG | Integer | Type of the computational grid:<br>= 1 When regular;<br>= 2 When irregular, but rectangular (not used);<br>= 3 When curvilinear. |
| STAGX | Real | Staggering of the curvilinear input grid with respect to the computational grid in X. |
| STAGY | Real | Staggering of the curvilinear input grid with respect to the computational grid in Y. |
| CVLEFT | Logical | The curvilinear computational grid is left/right-oriented. |
| RDTIM | Real | = 0 When in stationary mode;<br>= 1/DT When in non-stationary mode. |
| ICOND | Integer | Initial conditions:<br>= 0 When mode stationary, or no initial conditions needed;<br>= 1 When mode non-stationary and initial conditions should be calculated. |
| EXCFLD | Real | Exception values for input grids. |

| | | |
|---|---|---|
| NBFILS | Integer | Number of boundary condition files. |
| NBSPEC | Integer | Number of boundary spectra. |
| NBGRPT | Integer | Number of computational grid points for which boundary. |
| VARAST | Logical | Air-sea temperature difference is or is not variable over space. |
| BOTG(MCINGR) | Real | (Not used); Contains all of the items in /SWGRID/. BOTG is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.9 COMMON/ SWCOMG*

Location and dimensions of computational grid.

| Variable | Type | Description |
|---|---|---|
| ICOMP | Integer | Unused. |
| XPC | Real | X coordinate of the origin of the computational grid. |
| YPC | Real | Y coordinate of the origin of the computational grid. |
| ALPC | Real | Direction of x-axis of computational grid with respect to the user coordinates. |
| COSPC | Real | Cosine of ALPC. |
| SINPC | Real | Sine of ALPC. |
| XCLEN | Real | Length of computational grid in x-direction. |
| YCLEN | Real | Length of computational grid in y-direction. |
| MTC | Integer | Computational timesteps. |
| MXC | Integer | Grid points in x-direction of computational grid. |
| MYC | Integer | Grid points in y-direction of computational grid. |
| MDC | Integer | Grid points in the theta-direction of the computational grid. |
| MSC | Integer | Points in the sigma-direction of the computational grid. |
| SLOW | Real | Lowest spectral value of sigma. |
| SHIG | Real | Highest spectral value of sigma. |
| DX | Real | Mesh size in x-direction of computational grid. |
| DY | Real | Mesh size in y-direction of computational grid. |
| DDIR | Real | Mesh size in theta-direction of computational grid. |
| NX | Integer | Only used locally. Equal to MXS. |
| NY | Integer | Only used locally. Equal to MYS. |
| XCP | Real | Origin of the user coordinates with respect to the computational coordinates. |
| YCP | Real | Origin of user coordinates with respect to the computational coordinates. |
| ALCP | Real | Direction of user coordinates with respect to the computational coordinates. |
| DXRP | Real | Not used. |

| | | |
|---|---|---|
| DYRP | Real | Not used. |
| MSC4MI | Integer | Some counter for quadruplet interactions. Stored in WWINT(15). |
| MSC4MA | Integer | Some counter for quadruplet interactions. Stored in WWINT(16). |
| MDC4MI | Integer | Some counter for quadruplet interactions. Stored in WWINT(17). |
| MDC4MA | Integer | Some counter for quadruplet interactions. Stored in WWINT(18). |
| FRINTF | Real | Frequency integration factor (df/f). |
| FRINTH | Real | Frequency mesh boundary factor. |
| MMCGR | Integer | Grid points in computational grid. |
| FULCIR | Logical | Spectral directions cover full or part of circle. |
| SPDIR1 | Real | Represents the first spectral direction. |
| JSPDIR | Integer | Array *spcdir* within *pool* array. |
| JSIGMA | Integer | Array *spcsig* within *pool* array. |
| MCGRD | Integer | Number of wet grid points of the computational grid. |
| SPDIR2 | Real | Represents the second spectral direction. |
| IXCGRD | Integer | IX of the points of the computational stencil. |
| IYCGRD | Integer | IY of the points of the computational stencil. |
| KCGRD | Integer | Grid address of the points of the computational stencil. |
| XCGMIN | Real | Minimum x-coordinate of computational grid points. |
| XCGMAX | Real | Maximum x-coordinate of computational grid points. |
| YCGMIN | Real | Minimum y-coordinate of the computational grid points. |
| YCGMAX | Real | Maximum y-coordinate of the computational grid points. |
| NGRBND | Integer | Number of grid points on the computational grid boundary. |
| COMG (MCCOM) | Real | (Not used); Contains all of the items in /SWCOMG/. COM is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.10 COMMON/ SWNUMS*

Information related to the numerical scheme.

| **Variable** | **Type** | **Description** |
|---|---|---|
| NCOR | Integer | Not used. |
| IWCAP | Integer | Indicates whitecapping:<br>= 0 For command GEN1...;<br>= 0 For command GEN2...;<br>= 0 For command OFF WCAP..., no whitecapping;<br>= 1 For command GEN3 KOM...;<br>= 1 For command WCAP KOM ..., not documented in manual, standard WAM formulation (Komen et al., 1984); |

| | | |
|---|---|---|
| | | = 2 For command GEN3 JANS...;<br>= 2 For command WCAP JANS ..., not documented in manual, according to Janssen (1989, 1991);<br>= 3 For command WCAP LHIG ..., not documented in manual, according to Longuet-Higgins (1969), Yuan et al. (1986);<br>= 4 For command WCAP BJ ..., not documented in manual, according to Battjes and Janssen (1978);<br>= 5 For command WCAP KBJ ..., not documented in manual, combined formulation of Komen et al. (1984) and Battjes and Janssen (1978). |
| IPRE | Integer | Not used. |
| ICOR | Integer | Not used. |
| IBOT | Integer | Indicator bottom friction:<br>= 0 No bottom friction dissipation;<br>= 1 Set by command FRIC JON ..., JONSWAP bottom friction model;<br>= 2 For command FRIC COLL ..., Collins bottom friction model;<br>= 3 For command FRIC MAD ..., Madsen bottom friction model. |
| ICUR | Integer | Indicates presence of currents:<br>= 0 No currents;<br>= 1 For command READ CUR ..., currents are present. |
| IDBR | Integer | Not used. |
| IDIF | Integer | Not used. |
| IINC | Integer | Not used. |
| ITRIAD | Integer | Indicates triad interaction term:<br>= 0 Triads are inactive;<br>= 1 For command TRI DTA IMP ..., not documented in manual;<br>= 2 For command TRI DTA EXP ..., not documented in manual;<br>= 3 For command TRI [trfac] [cutfr], as in manual;<br>= 3 For command TRI LTA IMP ..., not documented in manual;<br>= 4 For command TRI LTA EXP ..., not documented in manual. |
| IREFR | Integer | Indicates refraction effect:<br>= 0 For command OFF REF, refraction is inactive;<br>= 1 Refraction is active. |
| ISURF | Integer | Indicates surf breaking (shallow water) term:<br>= 0 For command OFF BRE, surf breaking is inactive;<br>= 1 For command BRE CON ..., surf breaking with constant parameter; |

| | | |
|---|---|---|
| | | = 2 For command BRE VAR ..., surf breaking. |
| ITRSY | Integer | Not used. |
| IWIND | Integer | Indicates presence of wind and type of source term used:<br>= 0 No wind;<br>= 1 For command GEN1 ..., if wind is made active;<br>= 1 For command GROWTH G1 ..., not documented in manual, first generation source term;<br>= 2 For command GEN2 ..., if wind is made active;<br>= 2 For command GROWTH G2 ..., not documented in manual, second generation source term (as in Dolphin);<br>= 3 For command WIND ..., if IWIND still was 0, else unchanged;<br>= 3 For command GEN3 KOM ..., if wind is made active;<br>= 3 For command GROWTH G3 KOM ..., not documented in manual, third generation source term (Snyder);<br>= 4 For command GEN3 JANS ..., if wind is made active;<br>= 4 For command GROWTH G3 JANS ..., not documented in manual, source term by P. Janssen (1989, 1991);<br>= 5 For command GEN3 YAN ..., if wind is made active;<br>= 5 For command GROWTH G3 YAN. |
| IQUAD | Integer | Indicates the quadruplet interaction term:<br>= 0 For command OFF QUAD;<br>= 0 For command GEN1;<br>= 0 For command GEN2;<br>= 0 For command GROWTH G1;<br>= 0 For command GROWTH G2, quadruplets are inactive;<br>= 1 Quadruplets are calculated semi-implicit per sweep direction;<br>= 2 For command GEN3;<br>= 2 For command QUAD;<br>= 2 Set when *iwind* = 3 or 4 and *icur* = 0 in subroutine ERRCHK, quadruplets are calculated fully explicit per sweep direction;<br>= 3 Set when *iwind* = 3 or 4 and *icur* = 1 in subroutine ERRCHK, quadruplets are calculated fully explicit per iteration;<br>= iquad Set by command GEN3 ... QUAD [iquad]. |
| ICMAX | Integer | Number of points in computational stencil. |

| | | |
|---|---|---|
| ITERMX | Integer | Maximum number of iterations:<br>Set equal to MXITST for stationary computations.<br>Set equal to MXITNS for non-stationary computations. |
| NSTATC | Integer | Indicates stationary of computation:<br>= 0 Stationary computation;<br>= 1 Non-stationary computation. |
| NSTATM | Integer | = 0 Stationary mode;<br>= 1 Non-stationary mode;<br>= -1 Unknown. |
| U10 | Real | Wind velocity. |
| WDIP | Real | Wind direction with respect to problem coordinates. |
| WDIC | Real | PI2*((WDIP/PI2-NINT(WDIP/PI2)) |
| DEPMIN | Real | Threshold depth (to prevent zero divisions). |
| PWCAP | Real | Whitecapping coefficients. |
| PBOT | Real | Coefficients for the bottom friction models. |
| PTRIAD | Real | Controls the proportionality coefficient. |
| PNUMS | Real | Numerical coefficients. |
| PSURF | Real | Surf breaking coefficients. |
| PWIND | Real | Wind growth term coefficients. |
| SY0 | Real | Peak enhancement parameter of the JONSWAP spectrum. |
| SIGMAG | Real | Width of the Gaussian frequency spectrum in Hz. |
| ITFRE | Integer | Indicator for transport of action in frequency space:<br>= 0 For command OFF FSH, frequency shifting inactive;<br>= 1 Frequency shifting active. |
| NUMOBS | Integer | Number of obstacles. |
| LSETUP | Integer | = 0 Setup is not calculated;<br>= 1 Setup is calculated;<br>= 2 Setup is calculated with the boundary conditions from a nest file. |
| BNDCHK | Logical | Indicates whether computed Hs on boundary must be compared with the value entered as boundary condition. |
| HSRERR | Real | The error margin allowed between pre-scribed and calculated Hs at the upwave boundary. If exceeded, then a warning is produced. |
| FSHAPE | Integer | Indicates option for computation of frequency distribution in the spectrum (boundary spectra etc.). |
| DSHAPE | Integer | Indicates option for computation of directional distribution in the spectrum (boundary spectra etc.). |
| PSHAPE | Real | Coefficients for calculation of spectrum from integral parameters. |
| SPPARM | Real | Integral parameters used for computation of incident spectrum. |

| | | |
|---|---|---|
| BNAUT | Logical | Indicates whether Nautical or Cartesian directions are used. |
| ONED | Logical | Indicates whether the calculation should be performed in 1-D mode. |
| PQUAD | Real | Coefficients for quadruplet interaction. |
| BRESCL | Logical | Rescaling on/off. |
| IGEN | Integer | Indicates the generation mode:<br>= 1 For command GEN1;<br>= 2 For command GEN2;<br>= 3 For command GEN3. |
| PSETUP | Real | User defined level for correction of the setup. |
| CSETUP | Logical | Indicates whether or not the solver for setup has converged. |
| ACUPDA | Logical | Indicates whether or not action densities are to be updated during computation. |
| MXITST | Integer | Maximum number of iterations in stationary computations. |
| MXITNS | Integer | Maximum number of iterations in non-stationary computations. |
| NMS(MCNMS) | Integer | (Not used); Contains all of the items in /SWNUMS/. NMS is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.11 COMMON/ SWTEST*

Information for test output.

| Variable | Type | Description |
|---|---|---|
| LXDMP | Integer | Grid counter for a test point in the x-direction. |
| LYDMP | Integer | Grid counter for a test point in the y-direction. |
| NEGMES | Integer | Not used. |
| MAXMES | Integer | Not used. |
| TESTFL | Logical | Test output must/must not be made, mainly for test points. |
| NPTST | Integer | Number of test points; set by command TEST. |
| IPTST | Integer | Sequence number of a test point. |
| NPTSTA | Integer | Number of test points, equal to MAX(1, NPTST). |
| INTES | Integer | Testing parameter. |
| ICOTES | Integer | Minimum value for ITEST. |
| IOUTES | Integer | Minimum value for ITEST. |
| UNDFLW | Real | Small number to prevent underflows. |
| IFPAR | Integer | Unit reference number for output of parameters in test points. |
| IFS1D | Integer | Unit reference number for output of 1-D spectra of source terms. |

| | | |
|---|---|---|
| IFS2D | Integer | Unit reference number for output of 2-D spectra of source terms. If used, the value is made non-zero by subroutine FOR. |
| OUT(NKTST) | Real | (Not used); Contains all of the items in /SWTEST/. OUT is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.12 COMMON/ SWUITV*

Information for output.

| Variable | Type | Description |
|---|---|---|
| ALCQ | Real | Angle between x-axes of computational grid and output frame. |
| COSCQ | Real | Cosine of ALCQ. |
| SINCQ | Real | Sine of ALCQ. |
| IUBOTR | Integer | Set to one, when *ivtype* = 6 or 18. |
| INRHOG | Integer | Indicates the choice for output based on "variance" or "true energy".<br>= 0 Output based on variance;<br>= 1 Output based on "true energy". |
| ERRPTS | Integer | Unit reference number of file containing coordinates of "problem points". |
| DXK, DYK | Real | Mesh size of output frame. |
| ALPQ | Real | Angle between x-axes of user coordinate system and output frame. |
| COSPQ | Real | Cosine of ALPQ. |
| SINPQ | Real | Sine of ALPQ. |
| XQP | Real | X-coordinate (user coordinate) of origin of output frame. |
| YQP | Real | Y-coordinate (user coordinate) of origin of output frame. |
| XQLEN | Real | Length of x-side of output frame. |
| YQLEN | Real | Length of y-side of output frame. |
| OVSVTY | Integer | Type of the output variable:<br>= 1 Scalar;<br>= 2 Angle;<br>= 3 Vector;<br>= 4 Tensor;<br>= 5 Fully spectral quantity;<br>= 6 Directional spectral quantity. |
| OVLLIM | Real | Lower limit of validity of output quantity. |
| OVULIM | Real | Upper limit of validity. |
| OVLEXP | Real | Lower expected limit of output quantity. |
| OVHEXP | Real | Upper expected limit of output quantity. |
| OVEXCV | Real | Exception value for output quantity. |

| | | |
|---|---|---|
| SPCPOW | Integer | Power in expression for computation of average frequency. |
| AKPOWR | Real | Power in expression for computation of average wave number. |
| MXOUTAR | Integer | Calculates maximum memory needed for the output routines. |
| XPQ | Real | X-origin of a frame. |
| YPQ | Real | Y-origin of a frame. |
| OUTPAR | Real | Array containing various parameters for computation of output quantities. |
| UDA(MCUDA) | Real | (Not used); Contains all of the items in /SWUITV/. UDA is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.13 COMMON/ SWFYSP*

Physical parameters.

| Variable | Type | Description |
|---|---|---|
| GRAV | Real | Acceleration due to gravity. |
| WLEV | Real | Water level. |
| PI | Real | Circular constant. |
| PI2 | Real | 2*PI |
| RHO | Real | Density of the water. |
| DEGRAD | Real | Constant to transform degrees to radians. |
| DNORTH | Real | Direction of North with respect to the x-axis of user coordinates. |
| PWTAIL | Real | Coefficients to calculate the tail of the spectrum. |
| CASTD | Real | Air-sea temperature difference. |
| FP(MCFP) | Real | (Not used); Contains all of the items in /SWFYSP/. FP is used in a .for file; each item is listed individually in a .inc file. |

### *8.4.14 COMMON/ COMPDA*

Pointers for data arrays on computational grid.

| Arguments | Type | Description |
|---|---|---|
| JCMPDA | Integer | Array *compda* within *pool* array. |
| MCMVAR | Integer | Within array *compda*. |
| JHS | Integer | Significant wave height Hs within array *compda*. |
| JDISS | Integer | Dissipation within array *compda*. |
| JUBOT | Integer | Bottom orbital velocity within array *compda*. |
| JQB | Integer | Fraction of breaking waves within array *compda*. |
| JSTP | Integer | Steepness within array *compda*. |
| JDHS | Integer | Wave height correction within array *compda*. |

| | | |
|---|---|---|
| JDP1 | Integer | Old depth within array *compda.* |
| JVX1 | Integer | X of old current velocity within array *compda.* |
| JVY1 | Integer | Y of old current velocity within array *compda.* |
| JDP2 | Integer | New depth within array *compda.* |
| JVX2 | Integer | X of new current velocity within array *compda.* |
| JVY2 | Integer | Y of new current velocity within array *compda.* |
| JFRC2 | Integer | Friction coefficient within array *compda.* |
| JFRC3 | Integer | Friction coefficient within array *compda.* |
| JWX2 | Integer | X of new wind velocity within array *compda.* |
| JWY2 | Integer | Y of new wind velocity within array *compda.* |
| JBOT | Integer | Bottom level within array *compda*, not used. |
| JWLV1 | Integer | Old water level within array *compda.* |
| JWLV2 | Integer | New water level within array *compda.* |
| JWAREA | Integer | Work area within *pool* array. |
| JAC1 | Integer | Array *ac1* within *pool* array. |
| JAC2 | Integer | Array *ac2* within *pool* array. |
| JOUTD | Integer | Array *outda* within *pool* array. |
| JXYTST | Integer | Test points within *pool* array. |
| JTSTDA | Integer | Array *testda* within *pool* array. |
| MTSVAR | Integer | Within array *testda.* |
| JPWNDA | Integer | Within array *swtsda*, wind source term part A. |
| JPWNDB | Integer | Within array *swtsda*, wind source term part B. |
| JPWCAP | Integer | Within array *swtsda*, whitecapping. |
| JPBTFR | Integer | Within array *swtsda*, bottom friction. |
| JPWBRK | Integer | Within array *swtsda*, surf breaking. |
| JP4S | Integer | Within array *swtsda*, quadruplet interactions. |
| JP4D | Integer | Within array *swtsda*, quadruplet interactions. |
| JPTRI | Integer | Within array *swtsda*, triad interactions. |
| JAUX | Integer | Auxiliary array within *pool* array. |
| JDTM | Integer | Wave period correction within array *compda.* |
| MSWMAT | Integer | Within array *swmatr.* |
| JMATD | Integer | Within array *swmatr.* |
| JMATR | Integer | Within array *swmatr.* |
| JMATL | Integer | Within array *swmatr.* |
| JMATU | Integer | Within array *swmatr.* |
| JMAT5 | Integer | Within array *swmatr.* |
| JMAT6 | Integer | Within array *swmatr.* |
| JABIN | Integer | Within array *swmatr.* |
| JABLK | Integer | Within array *swmatr.* |
| JDIS0 | Integer | Within array *swmatr.* |
| JDIS1 | Integer | Within array *swmatr.* |
| JLEK1 | Integer | Within array *swmatr.* |
| JAOLD | Integer | Within array *swmatr.* |

| | | |
|---|---|---|
| JLEAK | Integer | "Leak" within array *compda*. |
| JWLV3 | Integer | Last read water level within array *compda*. |
| JVX3 | Integer | X of last read current velocity within array *compda*. |
| JVY3 | Integer | Y of last read current velocity within array *compda*. |
| JWX3 | Integer | X of last read wind velocity within array *compda*. |
| JWY3 | Integer | Y of last read wind velocity within array *compda*. |
| JDP3 | Integer | Last read depth within array *compda*. |
| JFL1 | Integer | Boundary spectra at time = T within *pool* array. |
| JFL2 | Integer | Boundary spectra at time = T + DT within *pool* array. |
| JAUXW | Integer | Auxiliary array within *pool* array. Used for WAM. |
| JAUXW2 | Integer | Auxiliary array within *pool* array. Used for WAM. |
| JAUXW3 | Integer | Auxiliary array within *pool* array. Used for WAM. |
| JFRW | Integer | Computed spectral frequencies WAM within *pool* array. |
| JANGSW | Integer | Computed spectral directions WAM within *pool* array. |
| JCOOX | Integer | X coordinates computational grid within array *compda*. |
| JCOOY | Integer | Y coordinates computational grid within array *compda*. |
| JADDRS | Integer | Indirect addresses of the computational grid within *pool* array. |
| JSETUP | Integer | Setup values within array *compda*. |
| JDPSAV | Integer | Saved depth (for setup) within array *compda*. |
| JWFRCX | Integer | Within array *compda*: x-computation is wave induced force. |
| JWFRCY | Integer | Within array *compda*: y-computation is wave induced force. |
| JUSTAR | Integer | Friction velocity within array *compda*. |
| JZEL | Integer | Roughness within array *compda*. |
| JTAUW | Integer | TauW within array *compda*. |
| JCDRAG | Integer | Drag coefficient within array *compda*. |
| JBFILS | Integer | Sequence number for pool array *bfiles*. |
| JBSPEC | Integer | Sequence number for pool array *bspecs*. |
| JBGRID | Integer | Sequence number for pool array *bgridp*. |
| JBSLOC | Integer | Sequence number for pool array *bsploc*. |
| JBSDIR | Integer | Sequence number for pool array *bspdir*. |
| JBSFRQ | Integer | Sequence number for pool array *bspfrq*. |
| JBSAUX | Integer | Sequence number for pool array *bspaux*. |
| JHSIBC | Integer | Significant wave height from boundary condition in array *compda*. |
| JGRBND | Integer | Pointer to *pool* array holding boundary grid. |
| JURSEL | Integer | *Ursell* number as used in Triad computation. |
| JASTD1 | Integer | Old air-sea temperature difference within array *compda*. |
| JASTD2 | Integer | New air-sea temperature difference within array *compda*. |
| JBTIME | Integer | Not used. |

| | | |
|---|---|---|
| JASTD3 | Integer | Last read air-sea temperature difference within array *compda.* |
| CDA(MCDA) | Real | (Not used); Contains all of the items in /COMPDA/. CDA is used in a .for file; each item is listed individually in a .inc file. |

## 8.5 (SWANOUT3 FOR FILE)

### *8.5.1 COMMON/ CPLT1(Not used)*

| Variable | Type | Description |
|---|---|---|
| IPLOT | Integer | Parameter specifying plot option IPLOT<br>= 0 No plotting of lines;<br>= 1 Plotting option on. |
| NN | Integer | Number of segments in which a basic line has to be divided. |
| LTEST | Integer | Parameter specifying quantity of test output of intermediate results. |
| IC1 | Integer | Number of steps after which the first number is plotted on a contour line. |
| IC2 | Integer | Number of steps between succeeding plot actions of a number on a contour line. |

## 8.6 (SWANPRE1 FOR FILE)

### *8.6.1 COMMON/ TIMFIL(Not used)*

Time related variables for the grids.

| Variable | Type | Description |
|---|---|---|
| INTECU | Integer | Timestep between non-stationary input conditions for currents. |
| INTEFR | Integer | Timestep between non-stationary input conditions for bottom friction. |
| INTEWI | Integer | Timestep between non-stationary input conditions for wind. |
| INTEWL | Integer | Timestep between non-stationary input conditions for water levels. |
| TBEGCU | Real | Start time for the non-stationary input conditions for currents. |
| TBEGFR | Real | Start time for the non-stationary input conditions for bottom friction. |
| TBEGWI | Real | Start time for the non-stationary input conditions for wind. |

| | | |
|---|---|---|
| TBEGWL | Real | Start time for the non-stationary input conditions for water levels. |
| TENDCU | Real | End time for the non-stationary input conditions for currents. |
| TENDFR | Real | End time for the non-stationary input conditions for bottom friction. |
| TENDWI | Real | End time for the non-stationary input conditions for wind. |
| TENDWL | Real | End time for the non-stationary input conditions for water levels. |
| TIMCU | Real | Last time that non-stationary input conditions has been read for currents. |
| TIMFR | Real | Last time that non-stationary input conditions has been read for bottom friction. |
| TIMWI | Real | Last time that non-stationary input conditions has been read for wind. |
| TIMWL | Real | Last time that non-stationary input conditions has been read for water levels. |

### *8.6.2 COMMON/ CBOUP(Not used)*

### *8.6.3 COMMON/ SWANWL*

Variables for project h3268.

### *8.6.4 COMMON/ TIMCOM*

Time related variables for the computation.

| Variable | Type | Description |
|---|---|---|
| TINIC | Real | Start time and date of the computation. |
| DT | Real | Timestep of the computation. |
| TFINC | Real | End time and date of the computation. |
| TIMCO | Real | Time and date of the computation during the simulation. |

### *8.6.5 COMMON/ TIMRED*

Time related variables for nested runs.

| Variable | Type | Description |
|---|---|---|
| BEGBOU | Real | Start time for the non-stationary boundary conditions. |
| TIMERB | Real | (Not used); Last time that non-stationary boundary conditions has been read in the case of nested runs. |
| IFACMX | Integer | Not used. |
| IFACMY | Integer | Not used. |

| | | |
|---|---|---|
| TINTBO | Real | Timestep between non-stationary boundary conditions in the case of nested runs. |

## 8.7 (SWANSER FOR FILE)

### *8.7.1 COMMON/ PSHAPE*

Coefficients of spectral distribution.

| Variable | Type | Description |
|---|---|---|
| PSHAPE(1) | Real | SY0, peak enhancement factor (gamma) in JONSWAP spectrum. |
| PSHAPE(2) | Real | Spectral width for Gauss spectrum in rad/s. |

### *8.7.2 COMMON/ SPPARM*

Array containing integral wave parameters.

| Variable | Type | Description |
|---|---|---|
| SPPARM | Real | Incident wave Parameters (Hs, Period, direction, Ms). |
| SPPARM(1) | Real | Hs, significant wave height. |
| SSPARM(2) | Real | Wave period given by the user (either peak or mean). |
| SSPARM(3) | Real | Average direction. |
| SSPARM(4) | Real | Directional spread. |

## 8.8 (OCPCOMM1 INC FILE)

### *8.8.1 COMMON/ REFTIM*

Origin for day and time.

| Variable | Type | Description |
|---|---|---|
| REFDAY | Integer | Day number of the reference day. The first day entered is used as reference day, the reference time is 0:00 of the reference day. |

## 8.9 (OCPCOMM3 INC FILE)

### *8.9.1 COMMON/ PLDATA*

Plotting related variables.

| Variable | Type | Description |
|---|---|---|
| IPLOPT | Integer | Plotting option. |
| IUPLF | Integer | Unit reference number of the PLOT file. |
| PLFACT | Real | Not used. |

| | | |
|---|---|---|
| PLPARM | Real | Plotting parameters. |

### *8.9.2 COMMON/ BINARY*

Common variables.

| **Variable** | **Type** | **Description** |
|---|---|---|
| BIT | Integer | Not used. |

## 8.10 (POOLCOMM INC FILE)

### *8.10.1 COMMON/ SWPOOL*

Data Pool.

| **Variable** | **Type** | **Description** |
|---|---|---|
| POOL | Integer | Dynamic data pool array. |
| RPOOL | Real | Real equivalence of *pool.* |
| LPOOL | Logical | Logical equivalence of *pool.* |

## 8.11 (SWCOMM2 INC FILE)

### *8.11.1 COMMON/ INPGRS (Not used)*

| **Variable** | **Type** | **Description** |
|---|---|---|
| IFLIDL | Integer | Lay-out in input file. |
| IFLIFM | Integer | Format identifier. |
| IFLNHF | Integer | Number of heading lines per file. |
| IFLNHD | Integer | Number of heading lines per input field. |
| IFLFAC | Real | Multiplication factor. |
| IFLNDS | Integer | Unit reference number of data file. |
| IFLNDF | Integer | Unit reference number of name list file. |
| IFLDYN | Integer | If = 0, Data is stationary,<br>If = 1, Non-stationary. |
| IFLTIM | Real | Time of last reading. |
| IFLBEG | Real | Begin time of data on file. |
| IFLINT | Real | Time interval of data on file. |
| IFLEND | Real | End time of data on file. |
| IFLFRM | Character | Format string. |

## 8.12 (SWCOMM4 INC FILE)

### *8.12.1 COMMON/ SWROP*

Higher order propagation and spherical coordinates.

| Variable | Type | Description |
|---|---|---|
| PROPSC | Integer | Indicates which numerical scheme is to be used for spatial propagation:<br>= 1 First order (BSBT);<br>= 2 SORDUP;<br>= 3 Third order (S&L). |
| PROPSL | Integer | Indicates which numerical scheme is used locally. |
| PROPSS | Integer | Indicates which numerical scheme is to be used in stationary computations:<br>= 1 First order (BSBT);<br>= 2 SORDUP. |
| PROPSN | Integer | Indicates which numerical scheme is to be used in non-stationary computations:<br>= 1 First order (BSBT);<br>= 3 Third order (S&L). |
| WAVAGE | Real | Indicates "wave age" parameter. |
| KSPHER | Integer | Indicates whether spherical coordinates are used, and which projection method:<br>= 0 Cartesian coordinates;<br>> 0 Spherical coordinates. |
| REARTH | Real | Radius of the earth. |
| LENDEG | Real | Length of a degree *ns*. |
| KREPTX | Integer | If > 0, the domain repeats itself in x-direction (primarily intended for propagation around the globe). |
| COSLAT | Real | Cosine of latitude;<br>= 1 for Cartesian coordinates. |
| PROJ_METHOD | Integer | Projection method:<br>= 0 (Quasi-)Cartesian;<br>= 1 Uniform Mercator (only spherical coordinates). |